License agreement

Agreement made between Prode "Prode" and "User".

• Prode is the owner of the product "Prode Properties" including, but not limited to, dynamic link libraries, static libraries, header files, sample programs, utility programs, together with the accompanying documentation collectively known as the "software".
• User desires to obtain the right to utilize the software, the parties hereby agree as follows

Personal license
A version with limited features is available for personal use at home or in educational establishments for teaching purposes, all other applications, without first obtaining a commercial license from Prode, are expressly prohibited.

Commercial license
Upon full payment of the license fee the User has full right to utilize the purchased number of units of the software, a unit is defined as one copy of the software or any portion thereof installed on one stand-alone computer, for networked computers one unit shall be applied for each user having concurrent access and one unit shall be applied for the server.

For all applications
• Prode grants the nonexclusive, nontransferable right to use the software.
• User has a royalty free right to reproduce and distribute the software as available from Prode Internet server (personal licence) provided that User doesn’t remove or alter any part of the software or of the licensing codes and threat the software as a whole unit.
• You cannot decompile, disassemble or reverse engineer the files containing the licensed software, or any backup copy, in whole or in part.
• You cannot rent, lease or sublicense the Licensed Software without express agreement by Prode.
• The software is provided "as is, where is", Prode does not warrant that software is free from defects, or that any technical or support services provided by Prode will correct any defects which might exist.
• Prode shall not be liable for any damages that may result directly or indirectly from the use of these software programs including any loss of profits, loss of revenues, loss of data, or any incidental or consequential damages that may arise out of use of these software.
• Your license is effective upon your acceptance of this agreement and installing the Licensed Software.
• This license agreement shall remain in effect until the Licensed Software will be in use.
• You may terminate it at any time by destroying the Licensed Software together with all copies. It will also terminate if you fail to comply with any term or condition of this Agreement. You agree upon such termination to destroy all copies of the Licensed Software in any form in your possession or under your control.

Customer support
Prode will provide the licensee with limited technical support by telephone, or by electronic media for a period of 60 days after delivery of the product.

How to contact Prode
you can contact Prode by phone, web page or email, the details are available at http://www.prode.com

How to obtain technical support
we welcome your comments or suggestions about our products, while the program has been tested carefully to ensure proper operation, it still may be possible for an unusual situation to result in an error. We will have a much greater chance of fixing or assisting with errors and problems if they are provided to us in a form that is repeatable.

In reporting a problem to us, the following information should be given:

• customer reference
• the version of the software
• a copy of the procedure you are running and if possible the input data
• a detailed description of what you were doing (sequence of operations) when the problem occurred
• any additional information you think may describe the problem
Introduction

Prode Properties includes a comprehensive collection of procedures to solve problems such as:

- Physical Properties Data
- Heat / Material Balance
- Process Simulation
- Process Control
- Equipment Design
- Separations
- Instrument Design
- And more ....

Technical features overview

Prode Properties library includes C++ ANSI ISO standard code with no limits on number of concurrent threads, the version for Windows (the same code is available for Android, Linux etc.) is released in form of compact Dynamic Libraries for direct access from Windows applications (Microsoft Excel, MATLAB, MathCad, Visual Studio applications including NET etc.).

- Support for Windows 7, 8, 10 (both 32 and 64 bit versions of library are included)
- Allows up to 500 different streams with up to 100 components per stream (user can redefine)
- Several compilations of chemical data and BIPs are available, the user can add new components and BIPs
- Comprehensive set of thermodynamic models
- Complete set of flash operations T-P, H-P, H-T, S-P, S-T, V-P, V-T, H-V, S-V, H-S, constant energy, phase-fraction...
- Functions for calculating specific properties of mixtures (critical point, Cricodentherm, Cricodenbar, cloud point etc.)
- Functions for calculating values and derivatives of fugacities, enthalpy, entropy, volume vs. temperature, pressure, composition
- Functions for calculating equilibrium lines at specified phase fractions (generation of phase diagrams)
- Functions for solving operating blocks as mixer, gas separator, liquid separator, distillation column, compressor, piping
- Functions for calculating stream properties as density, conductivity, viscosity (gaseous and liquid phases) surface tension, speed of sound, Joule Thomson etc.

Dynamic Link Libraries

A dynamic-link library is a binary file that acts as a shared library of functions that can be used simultaneously by multiple applications. These libraries are compatible with almost all Microsoft Windows applications and being compiled code they run very fast. They also integrate tightly with your application, allowing it to run as an autonomous program unit rather than being dependent on external modules of a different application.

Prode Properties (for Windows) distribution includes two libraries, ppp.dll (core methods) and pppx.dll (graphic interface).

Reference Literature

Although Prode Properties may appear easy to utilize also for people without a background in chemical engineering a basic knowledge in this area is useful for selecting the proper methods and critically evaluate the results, as general introduction to the matter we suggest:

- Introduction to Chemical Engineering Thermodynamics by Smith, Van Ness, Abbott
- The Properties of Gases & Liquids, by Reid, Prausnitz, Poling
- Phase Equilibria in Chemical Engineering by Walas
What's new

Release 1.1  [ 1994 ]
First distribution of Prode Properties as independent product, previously included in Prode Calculator, author Roberto Paron

Release 1.1c  [ 1997 ]
inclusion of multi-phase equilibria (vapor, liquid, solid, phases)

Release 1.2  [ 2003 ]
inclusion of several solvers for columns and reactors

Release 1.2a-d [2013-2019]
maintenance versions for porting to different platforms (Windows, Linux, Android, IOS)
<table>
<thead>
<tr>
<th>Features available vs. Versions</th>
<th>Personal</th>
<th>Base</th>
<th>Extended (**)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited number of components and features</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Database with 1650 (or 2300) chemicals</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Database with more than 30000 BIPs</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Complete set of thermodynamic models</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SAFT models</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Asphaltene models</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Wax models</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>GERG (2008)</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Electrolytes</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Hydrates Std. model</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Hydrates Complex model</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Derivatives vs. P,T,W of Fg, H, S, V</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Properties of fluids and mixtures</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Multiphase flash with specified T, P, H, S, V</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Additional flash operations</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Vapor-Liquid-liquid phase diagrams</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Vapor-Liquid-Solid phase diagram</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>VLE-LLE-SLE data regression</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Raw data regression utility</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Characterization of petroleum fractions</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Multiphase (gas,liquid) pipeline with heat transfer</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Isentropic nozzle HEM , HNE</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Isentropic nozzle HNE-DS , NHNE</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Polytropic stage, vapor-liquid (gas+liquid)</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Distillation (vapor-liquid)</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Distillation (vapor-liquid-liquid, liquid-liquid)</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Distillation batch (and dynamic)</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Depressuring unit (blow-down)</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
<tr>
<td>Reactions (different reactor types)</td>
<td></td>
<td></td>
<td>EX</td>
</tr>
</tbody>
</table>

(**) extended versions available with distribution license
Installing the program

Installation procedure
1) if previous versions are installed, login as admin and uninstall Prode Properties
2) download the last version of Prode Properties from this page, http://www.prode.com/en/download.htm
   and follow the instructions provided in the page

Obtain a commercial license
Your personal copy of Prode Properties doesn’t include all the features available with a full license, you can purchase a Software License (single installation) or a Network Based License

Order a software license
this license is based on the Installation Code which appears in Prode Properties Editor
• Install Prode Properties on your computer (see the description above)
• install the add-in for Microsoft Excel following the instructions in paragraphs “Getting Started from Microsoft Excel”
• Run Prode Properties Editor from Excel, select the License::Data page, in this example the ID is ADAC6J7D27779A3M
• order a software license and include your ID (to avoid errors you can copy and paste the text)

Activate a license
After the order, you will receive from Prode a text file containing a license key
• Run Prode Properties Editor from Excel, select the License::Data page and enter (to avoid errors you can copy and paste the text) your license key (see below), then click on button Store License

the software will report “License Key Stored”, at that point close Prode Properties Editor and stop Excel, the license will be activated the next time you’ll run Prode Properties
Prode Properties Quick Start

With Prode Properties you can solve complex problems with only minor programming effort. Much of the functionality is provided by the library. In this chapter you will learn step by step how to access Properties from your favourite application. This chapter is for those of you that want to skip the tutorial and immediately start using Properties. In the following sections, you will learn how to utilize the samples provided with Properties. When you run the samples you will get a broad overview of the possibilities available from using Properties

Locating the sample files
As default the sample files, including data files, project files, and other associated files are supplied with the program and placed in subdirectories under Prode main directory.

IMPORTANT

The installation procedure creates a directory \Prode\ and different subdirectories

\Prode\C includes definitions and code for C / C++ applications
\Prode\Excel includes samples for Microsoft Excel
\Prode\LIB includes the versions of the library
\Prode\MATLAB includes definitions and code for MATLAB applications
\Prode\MATHCAD includes definitions and code for MATHCAD applications
\Prode\Fortran includes definitions and sample code for Fortran applications
\Prode\NET\VBprops includes definitions and samples for Microsoft NET VB applications
\Prode\NET\C#-props includes definitions and samples for Microsoft NET C# applications

Data files folder

Prode Properties stores several files in a directory \Prode\ in user space, the exact path depends from Windows version and settings, in Windows 7, 8, 10 the folder is C:\ProgramData\prode

the list of files includes

chem.dat
pseudo.dat
bips.dat
mod.dat
def.ppp
res.lan
lic.dat
........

do not remove or rename these files, if Prode Properties cannot access these files (for example because they have been disseminated in different directories) an error message “Corrupted file, error reading data file” will be generated.

Make sure all users can access data files folder

IMPORTANT

When installing Prode Properties for users without full administrative rights make sure all users have read/write rights to data files folder,
if a user has no read/write rights on data files folder the program can generate errors and stop working.

Avoid errors in read / write operations

If a user doesn’t receive full read / write permissions on data files folder the program can generate a error when saving def.ppp or chem.dat files,
if you see this error you can
1) login as admin, and run Prode Properties
2) immediately before to save def.ppp or chem.dat (from Prode Properties) , with Windows File Manager manually delete the file which you wish to overwrite (def.ppp or chem.dat)
3) (from Prode Properties) save the file
Prerequisites
1) The different versions (32 or 64 bit) of Excel require different versions of Prode dll library, (Excel 32 requires Prode dll 32 bit while Excel 64 requires Prode dll 64 bit), when installing Prode Properties make sure to install the version suitable for your copy of Excel.
2) Before to run Prode Properties you should verify the separator in Regional Settings, by default Excel threats commas as separators and you should enter a macro as =EStrGD(1,300;1.0E5), if you wish to utilize a different separator, for example =EStrGD(1;300;1.0E5) you need to edit and modify the Regional Settings

Install Prode Properties add-in
before to use Excel you must load the add-in (file properties.xla) which instructs Excel about the methods included in Prode Properties library, you need to go through this procedure only once
In Excel 2010, 2013, 2016 open File menu, choose Options item and then Add-Ins

on the bottom select Manage Excel Add-Ins and click Go, you’ll see a list of add-ins, some checked, some not checked. If Prode Properties isn’t listed (and it won’t be unless you went through this procedure earlier) browse for the properties.xla file (by default installed in C:\Program Files\Prode\Excel) then back your way out.

Now Prode Properties should be listed in the list of add-ins, its box should be checked, click Ok to exit Excel Add-ins dialog, Prode Properties add-in provides instructions for Excel to access Prode methods and create a new menu.
Prode Properties Add-In menu

The menu for Prode Properties is available under Add-Ins tab in Microsoft Excel

Edit Properties : open Prode Editor
Open Archive : open existing files
Save a Archive : store data in Files

Working with archives

Prode Properties stores data in memory and you may wish to save in files so that the new information will not be lost when you end the program, each archive contains a copy of all streams (compositions, models, BIPs), plus units etc. read the paragraph "Working with archives, save and load data, default settings" for additional information
to Save data to archive, in Properties add-in menu select Save a Archive

then select the folder and the name of file

to Restore data, in Properties add-in menu select Open Archive

and select the archive to load
In Excel open a new page, File -> New, select a blank workbook, in add-in menu select Edit Properties to open Prode Properties Editor, there are different versions for Windows, Linux and Android, your version may include a tab selector or a tree menu (see herebelow), select Operating tab and stream 8

select Components tab and click Clear button (in case you see a composition defined), set units to molar flow and flow to 100 Kmol/h, from the first list select Methane as first component
then click Add button and define 0.6 as molar fraction for methane, continue and add ethane 0.2 mole fraction and propane 0.2 mole fraction

now we must define the models, goto Models tab and select PRX (PR Extended) in the list of predefined packages

this defines Peng Robinson Extended. for all properties (fugacity, enthalpy, entropy, volume) of vapor and liquid phases.
in Models dialog you can define the different options available for phase equilibria

<table>
<thead>
<tr>
<th>Multiphase equilibria</th>
<th>No multiphase, only two-phases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiphase initialization</td>
<td>Standard tests</td>
</tr>
<tr>
<td>Detect Phase State</td>
<td>From Globus or isothermal Compr. and Liq. Dens.</td>
</tr>
<tr>
<td>Phase diagram, check stability against feed</td>
<td>Accept all solutions</td>
</tr>
<tr>
<td>Phase diagram, specified phase fraction lines</td>
<td>End when crossing phase boundary lines</td>
</tr>
<tr>
<td>Hydrate structures inclusion</td>
<td>Include normal structures generated by formers</td>
</tr>
</tbody>
</table>

Multiphase equilibria instructs the procedure to calculate Vapor-Liquid, Vapor-Liquid-Liquid, Vapor-Liquid-Solid or Vapor-Liquid-Solid-Hydrate phase equilibria (for this stream)
Multiphase initialization and Detect phase state allow to reduce (in some cases) calculation time and improve identification of state in difficult cases
Phase diagram, check stability against feed and Phase diagram, specified phase fraction lines are specific for phase diagrams
Hydrate structures inclusion allows to include all the structures which may be generated by formers

You can edit / modify / define new thermo packages, to modify or create a new thermo package select a element in the list (for example 30), then define the models for the different properties and options, enter a name (for example Test) and finally click on Save button to store your package in memory

Next step, define BIPs
Many models available in Prode Properties require BIPs, Prode Properties stores BIPs in 4 different files (VLE, LLE, SLE and Hydrate).
To load BIPs from these files, in BIPs tab select VLE BIPs as data set and click on Get BIPs from Database to load BIPs.

Notice that each row requires to define C1 (first component in binary) C2 (second component) and one or more BIPs depending from selected thermodynamic model,
read "Binary Interaction Parameters (BIP)" and "Regress VLE-LLE-SLE data" for additional information about how to calculate BIPs,
store BIPs in database and retrieve BIPs.
Next step shows how to save stream data.
To save the stream, goto Operating tab, in first grid you can define a name for this stream, for example test and then click on Save button, the page now shows the new composition C1 0.6 C2 0.2 C3 0.2

From Operating tab it is possible to solve different operations, for solving an isothermal flash in second grid select TP-Flash (the operation to solve) stream 8 as feed (remember, results will be stored in stream selected in first row, in this case stream 8) define 200 K and 5 Bar.a as operating conditions and click on Compute button.

At these conditions Prode Properties (Peng Robinson Extended) calculates vapor+liquid equilibria.
Select the units:
most values require to specify unit, Prode Properties allows to define a set of predefined units,
in Prode Editor select Config and Units, define Bar.a as unit for pressure, K for temperature, Kg/m3 for density, KJ Kg/K for heat capacity

![Prode Properties Editor screen](image)

click on Ok button to leave Prode Properties editor.
Calling Prode Properties methods from Excel cells
if you are not interested goto “Getting Started from Microsoft Excel, part 3 working with predefined pages”

Once streams and units have been defined you can calculate different properties directly in Excel, we utilize the methods discussed in paragraph “Extended methods for accessing stream's properties”, these methods allows to calculate properties at specified conditions, you may wish to read the paragraph for additional information.

In B1 we enter 150 as temperature (remember we have K as unit) and in B2 we enter 5 as pressure (remember we have Bar.a as unit), the units of calculated values are Kg/m3 for density, and Kj Kg / K for heat capacity

in B3 enter the macro =EStrLf(8,B1,B2) for calculating liquid fraction of stream 8 at temperature specified in B1 and pressure specified in B2

in B4 enter the macro =EStrLD(8,B1,B2) for calculating density of liquid fraction,
in B5 enter the macro =EStrLcp(8,B1,B2) for calculating heat capacity of liquid fraction,
in B6 enter the macro =EStrGD(8,B1,B2) for calculating density of vapor fraction,
in B7 enter the macro =EStrGcp(8,B1,B2) for calculating heat capacity of vapor fraction.

Important : you may obtain different (more accurate) values with recent versions of software

In addition to the specific methods discussed in paragraph “Extended methods for accessing stream's properties”, with Excel you can utilize all the methods exported by Prode Properties library, the list includes methods to define streams, calculate a complete set of properties and solve complex operations such as columns, reactors etc.

With different methods there are different ways to define operating conditions, for example, if your units are K (temperature) and Bar.a (pressure) you can set 150 K and 5 bar.a as operating conditions in stream 1 with the macro

= setOp(8,150,5)

in the same way you can define the operating conditions as result, for example, of a H-P operation where you specify final pressure and enthalpy (in this case 15 Bar.and 500 kW)

= HPF(1,15,500,0)

we’ll see different applications in following examples
this example shows how to define operating conditions solving a isothermal flash in Prode Properties editor (as in previous example) and setOp() method. open Properties Editor and select the stream 8 defined in previous example with composition C1 0.6 C2 0.2 C3 0.2, as operation to solve select T-P VL (isothermal flash vapor-liquid), enter 170 K as operating temperature and 3 Bar.a as operating pressure and click on Compute to obtain the results.

Notice that you can obtain the same results with the macro

=setOp(8,170,5)

as previously examined.

in this example we adopt methods which do not include the values of temperature and pressure as input

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>150</td>
<td>=getT(8)</td>
<td>170</td>
</tr>
<tr>
<td>Pressure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Liquid Fraction</td>
<td>0.51733</td>
<td>0.40645</td>
<td></td>
</tr>
<tr>
<td>Liquid Density</td>
<td>576.8338</td>
<td>584.905</td>
<td></td>
</tr>
<tr>
<td>Liquid Heat Capacity</td>
<td>2.275305</td>
<td>2.214435</td>
<td></td>
</tr>
<tr>
<td>Gas Density</td>
<td>4.086361</td>
<td>3.734277</td>
<td></td>
</tr>
<tr>
<td>Gas Heat Capacity</td>
<td>2.134471</td>
<td>2.051359</td>
<td></td>
</tr>
</tbody>
</table>

in cell D1 enter =getT(8) to obtain operating temperature for stream 8
in cell D2 enter =getP(8) to obtain operating pressure
in cell D3 enter =StrL(8) to obtain liquid fraction
in cell D4 enter =StrLD(8) to obtain density of liquid fraction
in cell D5 enter =StrLcp(8) to obtain heat capacity of liquid fraction
in cell D6 enter =StrGD(8) to obtain density of vapor fraction
in cell D7 enter =StrGcp(8) to obtain heat capacity of vapor fraction
Next example shows how to define the conditions solving a H-P flash operation from Prode Properties editor.

Open Prode Properties editor, in operating tab, first row select stream (stream 8 defined in previous example with composition C1 0.6 C2 0.2 C3 0.2), as Operation to solve select H-P VL, as feed stream 8, the following two rows allow to specify input conditions (temperature and pressure) and output conditions (pressure) plus the heat to add or remove, we wish to model a control valve (no heat added / removed) with inlet conditions 270 K 50 Bar.a and outlet pressure 5 bar., enter values and click on Compute button to get the results.

The procedure predicts an output temperature of about 220 K with Vapor+Liquid.

As in previous example you can obtain equivalent results by entering the macros:

=HPF(8,5,EStrH(8,270,50),0)

where macro HPF solves the H-P operation with specified pressure 5 Bar.a and enthalpy calculated with macro EStrH(8,270,50)

we can now repeat the previous example to obtain the properties in Excel with new operating conditions.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>Temperature</td>
<td>220.8691</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Pressure</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Liquid Fraction</td>
<td>0.130289</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Liquid Density</td>
<td>570.6584</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Liquid heat Capacity</td>
<td>2.240764</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Gas Density</td>
<td>5.181223</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Gas Heat capacity</td>
<td>1.846163</td>
<td></td>
</tr>
</tbody>
</table>

in cell D1 enter =getT(8) to obtain operating temperature for stream 1
in cell D2 enter =getP(8) to obtain operating pressure
in cell D3 enter =StrLf(8) to obtain liquid fraction
in cell D4 enter =StrLd(8) to obtain density of liquid fraction
in cell D5 enter =StrLcp(8) to obtain heat capacity of liquid fraction
in cell D6 enter =StrGd(8) to obtain density of vapor fraction
in cell D7 enter =StrGcp(8) to obtain heat capacity of vapor fraction
Following a similar procedure it is possible to solve, from Prode Properties editor operations as separators and mixers. Differently from other operations, where the specifications define initial and final conditions, the mixer requires to specify the operating conditions for the two feeds before to solve the mixer operation, it is possible to define the operating conditions for the two feeds solving, for example, two isothermal flash operations. Supposing we wish to add stream 4 to stream 5 and obtain the results (new composition and operating conditions) in stream 9, we can follow this procedure:

1) select stream 4, define T-P VL operation with T = 300 P = 5 Bar.a click on Compute to solve
2) select stream 5, define T-P VL operation with T = 270 P = 2 Bar.a click on Compute to solve
3) select stream 9 (where the results will be stored), define Mixer, select streams 4 and 5 as feeds and specify 2 Bar.a as output pressure, click on Compute to solve

As in previous example note that you can obtain equivalent results with macro

=MIXF(9,4,5,2,0)

which solves a mixer operation, mixing streams 4 and 5 (results in stream 9) with specified output pressure 2 Bar.a. Also note that Mixer operation adopts the multiphase settings given for first stream (in this case stream 4), if you invert sequence

=MIXF(9,5,4,2,0) it shows a Vapor-liquid solution (no multiphase)
next example shows how to evaluate hydrate formation for a given mixture. Important: Hydrate models have a limited range of temperatures and pressures

Standard Hydrate Method 220 K < T < 310 K 0.1 Bar.a < P < 1000 Bar.a
Complex Hydrate Method 220 K < T < 320 K 0.1 Bar.a < P < 3000 Bar.a
the different operations can be solved only when t, p conditions are within the allowed range, outside this range the methods can return a "cannot converge calc’s loop" error meaning that procedure cannot calculate one (or more) point(s)

To calculate phase equilibria with hydrates you must include in stream one or more fomers plus water. In Properties Editor select stream "6 Test Hydrate", this stream includes a predefined composition C1 0.90375 C2 0.05 C3 0.02 CO2 0.02 H2O 0.005 CH4O 0.00125

When solving phase equilibria with solids and hydrates make sure to select the correct models for vapor, liquid, solid and hydrate phases, For hydrates Prode Properties Base includes a model based on PRXCPA-NRTL(P-HV) and one based on PRX-NRTL(P-HV), - PRXCPA-NRTL(P-HV) adopts PRXCPA-NRTL for vapor and liquid, SPRXCPA-NRTL for solid, HPRXCPA-NRTL for hydrate
- PRX-NRTL(P-HV) adopts PRX-NRTL for vapor and liquid, SPRX-NRTL for solid and HPRXCPA-NRTL for hydrate
in this example select the PRXCPA-NRTL(P-HV) package which automatically define the correct models

select as Multiphase equilibria option Multiphase Vapor-Liquid-Solid Hydrate
When calculating phase equilibria with solids (hydrates) to avoid large errors you must define BIPs, for this example select Hydrate BIPs as Data Set and click on Get BIPs from Database button to load the values.

now, in Components tab define the composition C1 0.906 C2 0.05 C3 0.02 CO2 0.02 H2O 0.004 CH4O 0

then, back to Operating tab and click on Save button to store the values in Prode Properties
Once saved you can calculate hydrate phase equilibria immediately selecting the TP-Flash operation, setting temperature (277 K) and Pressure (15 Bar.a), click on Compute button to see the results. At specified conditions the model indicates that hydrates can form.

You may decide to adopt methanol as inhibitor to avoid the formation of hydrates. We will consider a methanol (molar) fraction of about 1/4 or 0.00125 methanol vs. 0.0050 water. In component’s tab define the composition C1 0.90375 C2 0.05 C3 0.02 CO2 0.02 H2O 0.005 CH4O 0.00125.
in the Operating tab click on Save button to store the new composition
you can solve a TP-Flash operation to find the predicted hydrate formation pressure
in this case we test 277 K 30 Bar.a without finding hydrate formation.

In a similar way it is possible to simulate a valve with the H-P Flash operation, in this case (on the same stream 6) we simulate a valve with Tin 293.15 K Pin 50 Bar.a and Pout 15 Bar.a, the procedure calculates as final temperature about 275 K (without hydrate phase).
Prode Properties includes several methods for calculating equilibrium points, see the paragraph “Methods for thermodynamic calc’ s” for additional information.

LIPF() and LITF() as the name says are based on a liquid fraction specification, they returns the first point (along the specified liquid fraction line) at the specified pressure (or temperature).

PPIP() and PPTF() instead can accept a gas, liquid or solid fraction and calculate up to 5 points (at specified pressure or temperature) along the equilibrium line.

**double p = PttiF(integer stream, double t, double pf, int state, int n)***

which requires the stream, the equilibrium temperature, the phase fraction (range 0-1), the state (gas, liquid, solid) and the position (1-5) of the equilibrium point.

In cell B1 we define the temperature as 190.208 K, then in cells B40, B41, B42 we enter the macros:

- =PttiF(2,B1,0,1,1) in cell B2
- =PttiF(2,B1,0,1,2) in cell B3
- =PttiF(2,B1,0,1,3) in cell B4

where the first value (2) is the stream, the second (cell B1) represents the equilibrium temperature, the third (1) is the phase fraction (with 1 we specify 100% gas or a point on dew line, the same would be by setting the state as liquid and phase fraction as 0.0) the fourth (0) is the state (in Properties 0 = gas, 1 = liquid, 2 = solid) and the last is the required position (we require the points 1-3 along the dew line).

The procedure calculates the three equilibrium points (in this case units for pressure is Bar.a), if we change the temperature to 190.1 K we get different equilibrium pressures:

- File Home Insert Page Layout Formulas Data R e

Menu Commands

<table>
<thead>
<tr>
<th>B2</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>190.208</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>16.23916</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>30.04876</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>44.49123</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

the procedure calculates the three equilibrium points (in this case units for pressure is Bar.a), if we change the temperature to 190.1 K we get different equilibrium pressures:

- File Home Insert Page Layout Formulas Data R e

Menu Commands

<table>
<thead>
<tr>
<th>B2</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>190.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>15.87689</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>30.42524</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>44.30252</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IMPORTANT: do not enter Prode macros in Excel cells of predefined pages, instead, open a new blank workbook as discussed in “Getting Started from Microsoft Excel, part 2.”

Check the units, the input values in predefined pages are in K (temperature) and Pa (pressure) but you may change the units, open Prode Properties editor and set K for temperature and Pa for pressure or convert the input values to your preferred units.

Open the page multiphase.xls, (available in folder /prode/excel) this page allows to solve a multiphase equilibria problem and see the results in Excel.

Define stream 3 as feed, temperature 187 K and pressure 4154420 Pa.a (40 atm.g)

the procedure solves a isothermal flash showing the formation of two liquid phases.

In next example define stream 4 as feed, temperature 270 K and pressure 5000000 Pa.a (50 Bar.a) if required (different units selected) convert values.

The results show the presence of a vapor phase, a liquid phase (mainly hydrocarbons) and a solid phase (ice).
The page props.xls allows to calculate and graph tables of values in a range of temperatures for many different properties (liquid fraction, cp, cv, density, viscosity, thermal conductivity, speed of sound) for both gas and liquid phases (if present).

From Excel menu File->open , in Excel folder (in Prode Properties installation) select the file props.xls

If you wish you can modify the stream composition or the units of measurement (see previous examples).

In this page enter (in the proper units) the desired range of temperatures (cells B2-B3) and the operating pressure (cell B4) and click on compute button to calculate the data, Prode Properties will print the values with the desired units of measurement.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stream</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>t. min</td>
<td>1000 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>t. max</td>
<td>3000 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>pressure</td>
<td>100000 Pa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>tcalc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>T (K)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>100.00</td>
<td>1.00</td>
<td>2.40E+00</td>
<td>0.00E+00</td>
<td>1.90E+00</td>
<td>0.00E+00</td>
<td>3.09E-02</td>
<td>0.00E+00</td>
<td>8.02E-05</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>8</td>
<td>144.44</td>
<td>0.00</td>
<td>1.48E+00</td>
<td>2.11E+00</td>
<td>1.15E+00</td>
<td>1.55E+00</td>
<td>1.34E+00</td>
<td>1.05E+00</td>
<td>7.00E-05</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>9</td>
<td>190.44</td>
<td>0.00</td>
<td>1.48E+00</td>
<td>1.90E+00</td>
<td>1.79E+00</td>
<td>1.57E+00</td>
<td>1.17E+00</td>
<td>1.44E+00</td>
<td>1.77E+00</td>
<td>1.10E+00</td>
<td>7.51E-05</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>10</td>
<td>233.34</td>
<td>0.00</td>
<td>0.00E+00</td>
<td>1.17E+00</td>
<td>1.79E+00</td>
<td>1.13E+00</td>
<td>1.14E+00</td>
<td>1.56E+00</td>
<td>1.38E+00</td>
<td>9.01E-05</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>11</td>
<td>253.56</td>
<td>0.00</td>
<td>0.00E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
</tr>
<tr>
<td>12</td>
<td>277.78</td>
<td>0.00</td>
<td>0.00E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
</tr>
<tr>
<td>13</td>
<td>300.00</td>
<td>0.00</td>
<td>0.00E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
<td>1.17E+00</td>
</tr>
</tbody>
</table>

Note: you may need to load the add-in properties.xls to install the custom menu for properties calculator.

1) From Properties menu select the editor and define composition, model, options for the stream you wish to compute. You can also specify the units.

2) On this page specify operating pressure and a range of temperatures to compute the table.

Result: No errors.

Notes:
- Errors may be mitigated when accessing the chemical's data base with warning parameters.
- i.e., calculating liquid properties for temperatures below freezing point or above critical point etc., if you have errors within the temperature range.

19
20
21
22
23
24

Liquid fraction vs. temperature (click on rectangular)
The page phasenv.xls allows to calculate and graph a phase diagram (phase envelope) from Excel menu File->open, in Excel folder (in Prode Properties installation) select the file phasenv.xls. As for previous examples you can modify the stream composition, the units of measurement etc. from Prode Properties editor.

This phase diagram shows a three phase area for the mixture CH4 CO2 H2S (0.7 0.15 0.15)
To obtain different diagrams change the settings in models dialog

Multiphase equilibria option allows to calculate
1) vapor-liquid phase diagrams
2) vapor-liquid-liquid phase diagrams
3) vapor-liquid-solid phase diagrams

Phase diagram, check stability against feed, permits to test stability of calculated points against feed, unstable points are not printed, to show all calculated points change the settings.

Phase diagram, specified phase fraction lines, allows to end (or continue) lines after crossing a phase boundary, set to end (when crossing phase boundary lines) to avoid generating lines containing inconsistent data.

Next example will show a vapor-liquid-liquid phase diagram
a) enter stream 4, a predefined test case with a natural gas mixture including water
b) enter 0.0.5 as Phase Fraction and click on compute button to calculate phase diagram

Notice the water dew point line, the red line on the right
Next example will show a phase diagram with up to three dew points at the same temperature, enter 2 as stream, 0.01 as liquid fraction then click on button “Compute phase diagram” to generate the graph.

Observe that for this mixture the dew line (red line) shows up to three different equilibrium points at the same temperature (the area around 190 K), if you add the saturation point on the bubble line (black line) we have a total of four saturation point pressures at a given temperature, we will show how to calculate these points in Excel.
The page compressor.xls shows how to simulate a compression stage (as polytropic process) where the inlet stream can be vapor or vapor + liquid (mixed), comparing the results of different methods, see the paragraph “Methods for solving a Polytropic operation” for additional information.

From Excel menu File->open, in Excel folder (in Prode Properties installation) select the file compressor.xls

<table>
<thead>
<tr>
<th>Stream</th>
<th>Pin (inlet pressure)</th>
<th>Ti (inlet temperature)</th>
<th>Pout (outlet pressure)</th>
<th>Tout (discharging temperature)</th>
<th>Efficiency</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1000000.000 Pa.g</td>
<td>300.00 K</td>
<td>2000000.000 Pa.g</td>
<td>370.00 K</td>
<td>0.760</td>
<td></td>
</tr>
</tbody>
</table>

the page contains two sections, the first permits to calculate the polytropic efficiency of a single compression stage given the inlet temperature and pressure. The second section allows to estimate the discharging temperature given inlet temperature and pressure, outlet pressure and polytropic efficiency.

Notice that Prode Properties includes a specific method for solving a polytropic stage with phase equilibria, this method permits to simulate both single phase (vapor) and mixed (vapor + liquid) processes.

The mixture Methane 0.999, n-Butane 0.001 (predefined stream 2) at 10 Bar.a shows a dew point of 187.5 K, by setting an inlet temperature of 180 K we specify vapor + liquid as inlet condition, the standard method can simulate only gas streams, however the Polytropic solution with phase equilibria method allows to solve this case.
The page nozzle.xls allows to size a relief valve comparing the results of different methods for critical and two-phase flow, see the paragraph "Methods for solving a Isentropic operation" for additional information.

From Excel menu File->open , in Excel folder (in Prode Properties installation) select the file nozzle.xls

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stream</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Model</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Model parameter</td>
<td>0.7500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>T in</td>
<td>3.400E+02</td>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>P out</td>
<td>1.013E+05</td>
<td>Pa. a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Flow</td>
<td>1.2300</td>
<td>kg/s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Corrections Ka&quot;Kb&quot;K...</td>
<td>0.9000</td>
<td>0.3-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Estimated tout</td>
<td>274.7390</td>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Calculated area</td>
<td>4.229E-05</td>
<td>m2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Required Area</td>
<td>4.699E-05</td>
<td>m2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The steps to size a relief valve are easy to follow:
1) from Properties editor define the composition, models, BIPs (for mixtures)
2) enter the discharging temperature, pressure, flow, model, outlet pressure
3) click on button “Calculate Solution”

the procedure calculates the required area and the outlet temperature for critical and two-phase flow, you may utilize the procedure to verify the results from a different software in applications as fluids in critical area, two-phases flow etc.

The same page includes a procedure to compare the results from HEM (Homogeneous Equilibrium) and different Non Equilibrium models for a specified pressure in a range of inlet vapor qualities
Please follow these steps to compare two models,
1) from Properties editor define the composition, models, BIPs (for mixtures)
2) enter the pressure, model and parameter
3) click on button “Compare Models”

The Non Equilibrium models are mainly of interest for short nozzles where the final equilibrium condition (predicted by HEM models) is not reached cause the residence time of the fluid is too short.

The HNE models require specific parameters, for Prode HNE model a value of 0.75 is suggested for short nozzles but different values may be defined to fit specific data sets.
The page column.xls permits to solve a distillation column, refer to paragraph “Methods for solving staged columns” for additional information.

From Excel menu File->open , in Excel folder (in Prode Properties installation) select the file column.xls

In this page you can define different kind of columns with reboiler, condenser, one or more feeds and one or more side streams.

<table>
<thead>
<tr>
<th>SIMPLE STAGED COLUMN SIMULATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
</tbody>
</table>

The steps to define a column are easy to follow:
1) define the number of stages
2) define pressure distribution (bottom and top stage)
3) define stage efficiency
4) define the number of feeds, each feed flow rate and compositions (click on the proper Feed button to access the stream editor), each feed stage (remember that reboiler (if present) is stage 1 and condenser (if present) is stage N, and the liquid fraction (or the temperature) of each feed.
5) Define the number of side streams (if any), the stage, the type (vapor or liquid flow) and the flow specification
6) Define variables as condenser and reboiler and the related specifications, the procedure allows different specifications including molar fractions (and recovery) of a component in top or bottom stage

Notes:
In Stream Editor (Config->Units) you can define all the units for this project
in Stream Editor (Config->Settigss) you can define mass units or molar units for flows in Stream Editor
Once the column has been defined it is suggested to verify the input data for inconsistent specifications, if you are sure that all is Ok run the solver (button Solve Column)

**Error mass and energy balance**: -8.342E-16
**Reboiler duty**: 399.71527 kW
**Condenser duty**: 312.59773 kW

### Stage Temperatures, Pressures, Liquid and Vapor Flows in kmol/h

<table>
<thead>
<tr>
<th>Bottom product</th>
<th>T (K)</th>
<th>P (Pa.a)</th>
<th>LIQUID</th>
<th>C2H6</th>
<th>C3H8</th>
<th>C4H10</th>
<th>C5H10</th>
<th>C6H14</th>
<th>VAPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>263.16811</td>
<td>600000</td>
<td>29.7486</td>
<td>0</td>
<td>3.696E-08</td>
<td>0.543766</td>
<td>10.31693</td>
<td>18.988</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>285.80966</td>
<td>500612.2</td>
<td>29.7486</td>
<td>4.833105</td>
<td>17.99915</td>
<td>4.456998</td>
<td>2.460342</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>294.04396</td>
<td>501224.5</td>
<td>28.35777</td>
<td>0.158957</td>
<td>9.387879</td>
<td>9.311466</td>
<td>9.138249</td>
<td>9.7E-21</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>297.45478</td>
<td>501936.7</td>
<td>28.15443</td>
<td>4.509636</td>
<td>7.636268</td>
<td>9.23552</td>
<td>10.76431</td>
<td>9.02E-20</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>299.74269</td>
<td>503061.2</td>
<td>27.97346</td>
<td>0.394285</td>
<td>6.873339</td>
<td>8.402313</td>
<td>12.48684</td>
<td>6.65E-18</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>300.1505</td>
<td>503673.5</td>
<td>27.92731</td>
<td>4.26483</td>
<td>6.540123</td>
<td>8.052877</td>
<td>12.90783</td>
<td>5.44E-17</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>300.40397</td>
<td>504265.7</td>
<td>27.9659</td>
<td>4.24897</td>
<td>6.474781</td>
<td>7.789158</td>
<td>13.20706</td>
<td>4.57E-16</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>300.57011</td>
<td>504868</td>
<td>27.91466</td>
<td>4.24013</td>
<td>6.439535</td>
<td>7.597766</td>
<td>13.41315</td>
<td>3.75E-15</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>300.69484</td>
<td>505150.2</td>
<td>27.85995</td>
<td>0.43533</td>
<td>6.419117</td>
<td>7.406665</td>
<td>13.55561</td>
<td>3.06E-14</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>300.79124</td>
<td>506122.4</td>
<td>27.85021</td>
<td>4.233056</td>
<td>6.409866</td>
<td>7.368111</td>
<td>13.65403</td>
<td>2.55E-13</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>300.86972</td>
<td>506734.7</td>
<td>27.84206</td>
<td>0.423244</td>
<td>6.39665</td>
<td>7.293902</td>
<td>13.72177</td>
<td>2.03E-12</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>300.93596</td>
<td>507349.2</td>
<td>27.83998</td>
<td>4.232396</td>
<td>6.39541</td>
<td>7.253066</td>
<td>13.76811</td>
<td>1.65E-11</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>300.99376</td>
<td>507959.2</td>
<td>27.83793</td>
<td>4.234236</td>
<td>6.393393</td>
<td>7.221216</td>
<td>13.79595</td>
<td>1.34E-10</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>301.04574</td>
<td>508571.4</td>
<td>27.83635</td>
<td>4.236045</td>
<td>6.390326</td>
<td>7.199435</td>
<td>13.82024</td>
<td>1.08E-09</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>301.09365</td>
<td>509163.7</td>
<td>27.83599</td>
<td>4.230222</td>
<td>6.393254</td>
<td>7.184666</td>
<td>13.83417</td>
<td>8.74E-09</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>301.13972</td>
<td>509795.9</td>
<td>27.83608</td>
<td>4.240656</td>
<td>6.394374</td>
<td>7.174654</td>
<td>13.8428</td>
<td>7.06E-08</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>301.18118</td>
<td>510408.2</td>
<td>27.83662</td>
<td>0.424326</td>
<td>6.395974</td>
<td>7.166409</td>
<td>13.84791</td>
<td>5.7E-07</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>301.22351</td>
<td>511020.4</td>
<td>27.83473</td>
<td>4.246033</td>
<td>6.397904</td>
<td>7.163454</td>
<td>13.85056</td>
<td>4.59E-06</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>301.26426</td>
<td>511632.7</td>
<td>27.83333</td>
<td>4.248843</td>
<td>6.400042</td>
<td>7.161925</td>
<td>13.85145</td>
<td>3.7E-05</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>301.3046</td>
<td>512244.9</td>
<td>27.83269</td>
<td>0.425165</td>
<td>6.402148</td>
<td>7.160414</td>
<td>13.85076</td>
<td>0.00296</td>
<td></td>
</tr>
</tbody>
</table>

The report includes:
1) the verified errors in mass and energy balance
2) reboiler and condenser duties
3) temperature and pressure in each stage
4) total and component vapor flows in each stage
5) total and component liquid flows in each stage
The page hydrate.xls allows to calculate the hydrate formation curve directly in Excel. From Excel menu File->open, in Excel folder (in Prode Properties installation) select the file hydrate.xls

note that base version of Prode Properties allows to include only a few inhibitors (methanol, ethanol, MEG) with extended versions you can include salts and additional inhibitors
Getting started from LibreOffice

LibreOffice (and OpenOffice) include different methods for accessing external libraries as Prode Properties, LibreOffice BASIC allows to access Prode in Windows, while for Linux versions different interfaces (to Prode Properties) are available, there are several Libre Office pages in folder /Prode/LibreOffice, since LibreOffice (and OpenOffice) include similar features of Microsoft Excel, the LibreOffice pages look (and work) as the equivalent Excel versions, to open the LibreOffice pages, start LibreOffice Calc and Open the page phasenv.xls

you may receive a message “LibreOffice Security Warning” : The document contains document macros , click Enable Macros button and proceed

the page includes several buttons

Properties Editor opens the editor dialog
Open Archive opens a archive
Save Archive saves a archive
Compute phase diagram calculates the phase diagram for the specified stream
To calculate the phase diagram define a stream and click the button **Compute phase diagram**

in the same way you can load the pages for solving different problems as discussed in Excel section
Getting started from Python

The different versions of Prode Properties work with the different versions of Python available for Windows, Linux and Android platforms and all the methods exposed by Prode library can be imported in Python applications.

In Windows, to install Prode Properties library in your Python application, follow these steps:

1. Check whether your Python shell is executing in 32 or 64 bit mode, run Python and read the data (this is 64 bit)

   ![Python shell output]

   For a 32 bit Python application, copy the files from Prode\Python\32 folder.
   
   For a 64 bit Python application, copy the files from Prode\Python\64 folder.

   Copy the file prode.py to your Python application in /Lib folder.

   Copy the file _prode.pyd to your Python application in /DLLs folder.
to import prode module in Python, type

```python
>>> import prode
```
then you have access to all the methods exposed by Prode library, to run Prode Properties Editor on stream 1 enter

```python
>>> prode.edS(1)
```

to obtain the gas density at specified temperature and pressure (K,Pa units) for stream 1 enter

```python
>>> prode.setOp(1,220,2300000)
```
which returns 1 meaning the operation has been solved

```python
>>> prode.StrGD(1)
```
which returns the value (units Kg/M3) 31.269

with Python language you can create procedures to calculate and print phase diagrams, solve distillations columns etc. as discussed in Excel paragraph
Getting started with Microsoft NET (VB, C) applications

**IMPORTANT** Microsoft NET support files are located in the directory 'Prode\NET

Prode Properties can be easily included as unmanaged code in every Microsoft NET application, for compiling the sample code provided with Prode Properties a recent version of Microsoft Visual Studio is required. From Microsoft Visual Studio compiler menu File->Open->Project/Solution, in NET folder (in Prode Properties installation) select the file vba.sln

then from menu Build- select Build Solution. Note: if desired you can edit the settings from Project->vba Properties
As next step you can test the application, from Visual Studio menu Debug->Start Debugging, then once the application is running:

1) click on the button Prode Properties editor to access the editor, define the streams and units of measurement
2) define a suitable temperature and pressure (with proper units)
3) click on button Compute Properties to print the properties

you can then modify the code according your requirements.
Solving problems (introduction)

There are several different classes of problems which Prode Properties can help to solve but the most common are probably:

- physical properties of pure fluids and mixtures
- equipment design
- system simulation

Prode Properties provides many methods for the prediction of physical properties, in general a single instruction is required for calculating a property.

The design and rating of unit operations as distillation columns, towers, pumps, compressors, valves, heat exchangers etc. is another area where Prode Properties can result useful. The use of programming languages is generally suggested when dealing with complex problems while some formula in a worksheet can solve the usual work.

The system simulation may be used in the design stage to evaluate parameters, to help achieve an improved design or applied to existing systems for optimizing operating conditions. Generally the required solution is the list of operating conditions at the input and output of the operating blocks in the simulation block diagram. When there are no recycle streams or controls the method for solving the system is very simple: the output information from the first operating block is utilized as input for the second operating block and so on. However when there are output conditions which may interfere with input conditions some sort of iteration is required since some or all the equations governing the system may be non linear. There are two well known methods for solving such a system of non linear equations, the method of successive substitutions and Newton-Raphson, refer to good books of numerical analysis for additional information.

Streams

Most thermodynamic calcs in Prode Properties library take as reference a stream entity. For example when simulating a plant it makes sense to define different streams to represent flows in different sections, a stream usually defines compositions and operating conditions, Prode Properties supports a variable number of streams and most methods in Prode Properties require a reference to a stream, the reference is a numeric code (a progressive integer starting from 1 for first stream).

Streams attributes

As in process simulators each stream may include following information:

- a list of components and relative weights
- a value for the operating pressure
- a value for the operating temperature
- a value for the operating flow
- thermodynamic models for different properties
- a list of BIPs

Working with streams

Prode Properties permits to define complex topologies as there is no limit to the number of operating blocks required for simulating a plant, with Prode Properties for simulating a plant you convert the different sections into pieces of code, to do so you can use the basic blocks available in all process simulators, for example:

- isothermal flash, for calculating multiphase equilibria at the specified temperature and pressure
- flash unit (enthalpy, entropy or volume basis), calculates output temperature or pressure, with this unit you can simulate pipelines, valves, heat exchangers, pumps, compressors and many others operations.
- fixed vapor fraction flash, for constructing phase envelopes, calculating bubble and dew points etc.
- mixer to add the contents of two streams
- divider to subtract a part of flow from a stream

by putting together these blocks it is possible to simulate also complex plants.

Simulating a plant

Transform the flow sheet in a simulation block diagram, fluid and energy flow diagrams are standard engineering tools, you assign a number to the different streams and identify the basic blocks which will be solved by Prode Properties.

Notice the number which identifies each stream, in this case different numbers mean (possible) different compositions (we do not consider chemical reactions here), the output of each block can be easily calculated providing the input has been defined.
Working with archives, save and load data, default settings

Load and save archives
Archives are files which contain the data required by Prode Properties to work with stream's and units of measurement, when you open an archive the stream's data and units are loaded, when you choose to save an archive these data are stored in a file, in this way you can work with many different projects.
Prode Properties includes several methods to save and load data as archives.

The default settings
When Prode Properties starts it loads data from the archive named "def.ppp" so if you wish to use your own list of streams, units etc. just save your preferred settings under the name "def.ppp".

Properties editor

Prode Properties includes Properties editor, from the editor you have access to
- Streams edit operating conditions, flow, compositions, models, BIPs for all streams
- Config edit all units of measurement and settings
- Chemicals edit all chemical's data, regress data, add new chemicals
- BIPs edit BIPs, regress data, add new BIPs

How to activate the Properties editor
call one of the methods edS(stream), edSS() remember that in Prode Properties each stream is referenced with a code (integer value) in the range (1... max number of streams).

From Microsoft VB

Call edSS()
Call edS(8) ' start editing stream nr. 8

From Microsoft Excel

=edSS()
=edS(8) ' start editing stream nr. 8

From Microsoft Visual C++

edSS();
edS(8); // start editing stream nr. 8
Stream operating

From this page you can:
- select a stream (select / edit stream)
- define a name for the selected stream
- save edited stream data
- Compute different flash operations

Define a new stream / edit existing streams
- select the stream (you wish to edit / define) from the “Select / edit stream list”
- go to page “Components”, define the list of components and relative weights
- go to page “Models”, define the models for the different properties, define settings, load BIPs
- back to page “Operating”, put a name for the stream and click on “Save”
- check that correct stream composition appears on the second grid

Compute a flash operation
- make sure a feeding stream has been properly defined (composition, models, settings) then select the feeding stream
- select the flash operation
- define the required specifications and click on “Compute”
### Predefined Operations

Prode Properties (Base) allows to solve directly from editor a few operations (see paragraph "Methods for thermodynamic calc’s" for the list of methods available in library), these methods can solve multiphase equilibria (vapor-liquid-solid phases)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-P Flash</td>
<td>Temperature, Pressure, Flash Operation</td>
</tr>
<tr>
<td>LF-P</td>
<td>Phase Fraction, Pressure, Flash Operation</td>
</tr>
<tr>
<td>LF-T</td>
<td>Phase Fraction, Temperature, Flash Operation</td>
</tr>
<tr>
<td>H-P Flash</td>
<td>Enthalpy, Pressure, Flash Operation</td>
</tr>
<tr>
<td>H-T Flash</td>
<td>Enthalpy, Temperature, Flash Operation</td>
</tr>
<tr>
<td>S-P Flash</td>
<td>Entropy, Pressure, Flash Operation</td>
</tr>
<tr>
<td>S-T Flash</td>
<td>Entropy, Temperature, Flash Operation</td>
</tr>
<tr>
<td>V-P Flash</td>
<td>Volume, Pressure, Flash Operation</td>
</tr>
<tr>
<td>V-T Flash</td>
<td>Volume, Temperature, Flash Operation</td>
</tr>
<tr>
<td>Copy Stream</td>
<td>copy a stream into another stream</td>
</tr>
<tr>
<td>Gas Sep.</td>
<td>simulates a gas separation at specified temperature and pressure</td>
</tr>
<tr>
<td>Liquid Sep.</td>
<td>simulates a liquid separation at specified temperature and pressure</td>
</tr>
<tr>
<td>Mixer</td>
<td>mix two streams</td>
</tr>
</tbody>
</table>
Stream Composition

From this page you can:
• define a list of components by selecting components from the library
• specify the amount of each component.

Define the sorting criteria
• select the preferred criteria

Add a component to the list
• select a component from the list of components
• click on Add button

Remove a component from the list
• click on Remove button to remove the last component in the list

Clear the list
• click on Clear button to clear all components in the list

Units
• select the desired Mole or Weight fractions (and flow)

Define flow (stream)
• enter flow (stream)

Define reactions sets (for reactive flash operations)
• select the reaction set and define reactants and products (for the selected components)

Balance chemical equations
• select a reaction set, define products and reactants, then click on Balance button to balance chemical equations
Stream Models

From this page you can:
- define many different packages with user defined models and options
- set models per each property and state (gas, liquid, solid)
- set different options

Define a new package / Edit existing package
- select a package in the list
- in Models and Options Window select the models and options for this package
- if required edit / change the name for the package,
- click on "Save" button to save this package

Select a package
- select a package in the list

Define models
- specify the models per each property and state

Set a option
- Multiphase equilibria, allows to define different solutions as vapor-liquid, vapor-liquid-liquid and vapor-liquid-solid
- Multiphase initialization, allows to reduce the number of trial phases thus reducing time required
- Detect Phase State, allows to use different methods to detect the state of each phase
- Phase diagram, check stability against feed, allows to include stability analysis on each calculated point
- Phase diagram, specified phase fraction lines, allows to terminate lines when crossing a phase boundary
- Hydrates structures inclusion, allows to test all possible hydrate structures which may be generated by former(s)
BIPs

From this page you can:
• input / edit / load BIPs for the different models

Define a list of BIPs
• click on Get BIPs for loading all BIPs available in library
• if required add your own specific BIPs

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>U1-2</th>
<th>U2-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>538 578</td>
<td>354 352</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

IMPORTANT
For BIPs the first two columns Ci and Cj define the component’s position in the list (i.e. 1 for the first component, 2 for second and so on), while other columns allow to enter the values for the different BIPs required by selected model, note that some values (for example in Wilson, NRTL etc.models) have units of cal / gr.mol.
Example, binary of water methanol UNIQUAC U12 : 538, U21 : -354 in the first two columns (c1, c2) enter the components relative position in the list, assuming that water is the first component and methanol the second

<table>
<thead>
<tr>
<th>c1</th>
<th>c2</th>
<th>U1-2</th>
<th>U2-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>538</td>
<td>-354</td>
</tr>
</tbody>
</table>

IMPORTANT
Prode Properties allows to define Temperature Dependent BIPs for many models, Temperature Dependent BIPs can provide additional accuracy.
From this page you can:

- define the unit of measurement for the different properties

**IMPORTANT**

With Prode Properties you have complete control over the engineering units, this means that the program permits to select from a list of different units for each property, and Prode Properties automatically converts the input values and the results according to the selected units.

1) if you set Bar.a as unit for pressure all inputs and outputs will be in that unit
2) the new units will become effective after closing the Editor (select Ok button to confirm the Units)
3) if you wish to define your own set of units remember, before to leave the application, to save data into a archive otherways your changes will be lost
Config Settings

From this page you can:
- define the different settings as max number of streams, the temperature and pressure at reference conditions, the base values for enthalpy and entropy, convergence tolerance etc.

Configurable parameters:
- max number of streams
- max number of components per stream
- max number of interaction coefficients pairs per stream
- reference temperature and pressure
- base values for enthalpy and entropy calc's
- convergence tolerance
- max allowed time for solving a operation
- Flow units
- minimum liquid density to validate liquid phase

IMPORTANT
before to leave the application remember to save data into the archive differently your changes will be lost
From this page you can:
- edit and change the physical properties data included in the databank
- save all data in a file

**Edit / modify data:**
- select a component from the component’s list
- edit / modify the related fields (see also the section with description of all fields)
- select the “Save” button to save the modified data (differently new data will be discharged)

**Adding a new component:**
- select the “New” button
- edit the related fields (see also the following page with description of all data fields)
- select the “Save” button to save the data (differently new data will be discharged)

**Remove a component:**
- select a component from the component’s list
- select the “Remove” button

**IMPORTANT**

**Updating the file which contains the databank:**
this option permits to store all data into a file, differently all changes will be lost when leaving the application
- select the “File” button

**CAUTION:** you may wish to create a backup of the file chem.dat before to overwrite the file
Chemicals Settings

From this page you can:

- edit informations (CAS number) required to identify some components.
Regress raw data

From this page you can:

• regress raw data into values compatible with chemical’s database

Regress raw data

• in Chemicals Data page select a chemical
• in Chemical Regress page select a property and the correlation for fitting raw data
• enter the available data (all temperature and value pairs) with the proper units of measurement
• if required you may enter initializing values and limits for calculated parameters
• click on Calculate button, evaluate calculated values and errors, you may try different correlations for best data fitting
• click on Transfer button to copy calculated values into Chemicals Data page
• in Chemicals Data page select the “Save” button to save the data (differently new data will be discharged)

IMPORTANT

Prode Properties flexible database format supports more than 15 different correlations (30 in extended version), so for each property you can select the correlation which best fits experimental data.
From these pages you can:
• edit Binary Interaction Parameters
• add / remove Binary Interaction Parameters
• regress VLE (vapor-liquid), LLE (liquid-liquid), SLE (solid-liquid) data points
• save all data in a file

Edit / modify data:
• select two components from the component's lists
• select the database (VLE/LLE/SLE/Hydrate)
• select the model
• edit / modify BIPs
• select the "Save" button to save the modified data (differently new data will be discharged)

IMPORTANT

Updating the file which contains the BIPs data:
this option permits to store all data into a file, differently all changes will be lost when leaving the application
• select the “File” button

CAUTION you may wish to create a backup of the file bips.dat before to overwrite the file
Regress VLE-LLE-SLE data

From this page you can:

• regress VLE-LLE-SLE data for calculating the best fitting parameters for different models.

as first step in BiPs->Data page you must select two components (in this example methanol, first component, and water).

The procedure permits to enter experimental (measured) VLE-LLE-SLE data points or fit automatically the model to VLE points calculated with UNIFAC.

• select the models for the different states (in this case PRX-NRTL-HV)
• select measured VLE-LLE-SLE data points as data to regress
• select Standard as solution mode, this option allows to calculate all the BiPs for specified model
• select $F = xerr*yerr$ as minimization mode, this is the default

enter the measured VLE (vapor-liquid data points)
on each row include in X1 column the measured liquid molar fraction of component 1, in Y1 column the measured vapor molar fraction of component 1, for LLE (liquid-liquid equilibria) include in X1 column the measured liquid molar fraction of component 1 in first liquid phase, and in Y1 column the measured liquid molar fraction of component 1 in second liquid phase, for SLE (solid-liquid equilibria) include in X1 column the measured liquid molar fraction of component 1 in liquid phase, and in Y1 column the measured solid molar fraction of component 1, finally enter the temperature and the pressure for that point.

Example of measured VLE data for Methanol-Water

VLE data for Methanol-Water at 735 mmHg

<table>
<thead>
<tr>
<th>Point</th>
<th>X(1)</th>
<th>Y(1)</th>
<th>Temperature (C)</th>
<th>Pressure (mmHg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLE</td>
<td>0.008400</td>
<td>0.103000</td>
<td>96.5</td>
<td>735</td>
</tr>
<tr>
<td>VLE</td>
<td>0.025800</td>
<td>0.227000</td>
<td>92.3</td>
<td>735</td>
</tr>
<tr>
<td>VLE</td>
<td>0.068000</td>
<td>0.391000</td>
<td>87.5</td>
<td>735</td>
</tr>
<tr>
<td>VLE</td>
<td>0.137000</td>
<td>0.568000</td>
<td>80.1</td>
<td>735</td>
</tr>
<tr>
<td>VLE</td>
<td>0.240000</td>
<td>0.680000</td>
<td>75.9</td>
<td>735</td>
</tr>
<tr>
<td>VLE</td>
<td>0.480000</td>
<td>0.790000</td>
<td>70.6</td>
<td>735</td>
</tr>
<tr>
<td>VLE</td>
<td>0.572000</td>
<td>0.820000</td>
<td>68.7</td>
<td>735</td>
</tr>
<tr>
<td>VLE</td>
<td>0.741000</td>
<td>0.906000</td>
<td>66.4</td>
<td>735</td>
</tr>
</tbody>
</table>
• in Data dialog select the two components, in this example first is methanol, second is water
• in Regress dialog:
  • select the models for the vapor and liquid phases (in this case PRX-NRTL-HV)
  • select measured VLE-LLE-SLE data points as data to regress
  • select Standard as solution mode, this option allows to calculate all the BIPs for specified model
    if you do not select the Standard solution mode the procedure shows (for PRX-NRTL-HV model) only one BIP
    which is $K_{ij}$ in Peng-Robinson EOS , in this way you can calculate both sets
  • select $F = x_{err} \cdot y_{err}$ as minimization mode, this is the default
  • enter the measured data (see previous page) selecting the proper units

Selecting the button Calculate the procedure calculates the best fitting parameters, shows the calculated results and the
relative errors (%)
you can easily compare the different models
as alternative the procedure offers the option to calculate the best fitting parameters to VLE data points calculated with UNIFAC, this entirely predictive method doesn’t require experimental data and in some cases can result useful. Select “VLE points calculated with UNIFAC” in “Regress” selector to use this method.

Once the parameters have been calculated you must select the “Save” button to transfer the results in memory, from BIPs->Data page you can then inspect the values and store permanently in file (to store in the file select File button).
Regress SLE (Water-Methanol) and test the calculated freezing point depression
This example will show how to estimate BIPs for different models from available SLE equilibrium points

SLE data for Water-Methanol, atmospheric pressure
in this example first component is Water, second component Methanol
X1 is the water fraction in liquid phase while Y1 is solid fraction (we can set 1 for a solid pure model)

<table>
<thead>
<tr>
<th>Point</th>
<th>X1</th>
<th>Y1</th>
<th>Temperature (K)</th>
<th>Pressure (Bar.g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLE</td>
<td>0.9432</td>
<td>1</td>
<td>266.85</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>0.9</td>
<td>1</td>
<td>259.65</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>0.87676</td>
<td>1</td>
<td>257.65</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>0.80583</td>
<td>1</td>
<td>246.85</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>0.8</td>
<td>1</td>
<td>242.95</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>0.728</td>
<td>1</td>
<td>233.45</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>0.641</td>
<td>1</td>
<td>217.95</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>0.636</td>
<td>1</td>
<td>214.95</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>0.6</td>
<td>1</td>
<td>208.15</td>
<td>0</td>
</tr>
</tbody>
</table>

as first step in BIPs->Data page you must select the components, first component is water, second component is methanol, then select the database (SLE BIPs) where to store calculated BIPs and the model PRX-NRTL-HV (Peng-Robinson + NRTL with Huron Vidal mixing rules)

in BIPs->Regress page select models (PRX-NRTL-HV for vapor, PRX-NRTL-HV for liquid, SP-PRX-NRTL-HV for solid) as for previous example select Standard as solution mode, which allows to calculate all the BIPs for the specified model, and enter all data points, click on Calculate button to obtain the results

click on Save button to move calculated values in previous page

In BIPs->Data page verify values (see below), select SLE BIPs Database and click on Save button to store data (you can store in file with the File button)
Now we wish to test the accuracy of calculated BIPs in estimating freezing point depression. In Prode Properties Editor select stream 9 and define as composition C1 0  C2 0 CH4O 0.364 H2O 0.636 to test this point in the series

<table>
<thead>
<tr>
<th>SLE</th>
<th>X1</th>
<th>Y1</th>
<th>T(K)</th>
<th>P(Bar.g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.636</td>
<td>1</td>
<td>214.95</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
In Models tab select the same models adopted in data regression, (PRX-NRTL-HV for vapor, PRX-NRTL-HV for liquid, SP-PRX-NRTL-HV for solid) you may select the PRX-NRTL-HV package.

In BIPs tab select SLE BIPs Data set (the database where calculated BIPs have been stored) and click on Load BIPs button to get BIPs, make sure that values shown on window are those previously calculated.

In Operating tab click Save Button to define the stream and make sure the list of components has been updated. Now we can solve a TP-VLS flash operation to evaluate the point of incipient solidification.

The calculated point is about 214.5 K, compare this value with the experimental data (214.95 K) utilized in data regression, error is lower than 1 K.
With the same method it is possible to estimate solid formation for mixtures of different fluids, for example water and hydrocarbons with methanol as inhibitor, we repeat the example including C1 and C2 and keeping constant water / methanol fraction, define the mixture C1 0.87 C2 0.12 H2O 0.00636 CH4O 0.00364 for this mixture the methanol fraction in water is equivalent to previous example, at 10 Bar the estimated point for solid (ice) formation is about 214.5 K (as previously calculated).

Note that we have not included BIPs for water-hydrocarbons and methanol-hydrocarbons, for accurate results you may wish to include BIPs for all pairs. The PR-NRTL-HV model included in Prode Properties allows to enter (for each pair) a set of three BIPs (BIP1, BIP2, BIP3) or a single BIP (BIP4), when only single BIPs (BIP4), are included PR-NRTL-HV model gives the same results of PRX model.
Parameters of models

From this page you can:
• edit the parameters required by the different models available in library

**Edit / modify data:**
• select the components from the component's lists
• select the model
• edit / modify the parameters
• select the “Save” button to save the modified data (differently new data will be discharged)

**IMPORTANT**

**Updating the file which contains the Model data:**
this option permits to store all data into a file, differently all changes will be lost when leaving the application
• select the “File” button

**CAUTION** you may wish to create a backup of the file bips.dat before to overwrite the file
Accessing Prode Properties library

The technique for accessing the methods available in Prode Properties library will depend on which programming language you use. Languages such as FORTRAN, C, C++ or Microsoft NET (VB,C) exhibit differences in parameter passing in and out of functions. This may require you to adapt your code from the examples shown here. The calling convention determines how a program makes a call and where the parameters are passed. PROPERTIES does use of standard calls of Windows API, it pushes parameters on the stack, in reverse order. When accessing PROPERTIES you must also consider:

- Prode Properties real type is 8 bytes
- Prode Properties integer type is 4 bytes
- Parameters are passed by value (with exception of strings which are arrays of characters)

**IMPORTANT**

C / C++ support files are located in the directory \\Prode\C
FORTRAN support files are located in the directory \\Prode\FORTRAN
Microsoft NET support files are located in the directory \\Prode\NET
Microsoft EXCEL support files are located in the directory \\Prode\Excel

**Fortran**

add ppp.lib file to the list of the files in Your project and include ppp.f90 to instruct the compiler about the methods available in Prode Properties then access the methods as they were included in your code

```
C this procedure returns the critical temperature of a compound
INTERFACE TO REAL*8 FUNCTION TC ([C,ALIAS:'CompTc'] comp)
INTEGER*4 comp [VALUE]
END

REAL*8 tc
INTEGER*4 id
C define the id value here
tc = TC(id)
```

**C / C++**

- include the ppp.h header
- add ppp.lib file to the list of the files in Your project
- make sure you use the calling convention of ppp.h header file,
- access the Prode Properties methods

```
char *name;
name = CompN(1); // returns the name of the first component in the chemical's file
```

**Microsoft NET (VB)**

see the samples provided with Prode Properties for additional information

- include properties.vba to instruct the compiler about the methods available in Prode Properties and access the methods

```
CompName = MCompN(1)
```

**Microsoft Excel**

Microsoft Excel supports macros and VBA language for defining procedures; both of these can be used to access the functions in Prode Properties, see the samples provided with Prode Properties for additional information

- include properties.vba to instruct Microsoft Excel about the methods available in Prode Properties and access the methods

```
CompName = MCompN(1)
```
Translate resources to different languages

A large part of the resources are stored in the file res.lan, see the paragraph “Data files folder” for additional information about how to locate the file. The file res.lan is a text file, easily editable by the user.

Example

in English language
N2_NAME = “Nitrogen”;

in French language
N2_NAME = “Azote”;

in Italian language
N2_NAME = “Azoto”;

IMPORTANT

When editing a string take care to modify only the parts enclosed within the braces “” and do not alter/modify the data structures composed by special characters as for example ::

Microsoft Applications and Strings

Prode Properties utilizes the standard API calling convention for Microsoft Windows applications. This assures that almost all Windows compatible applications which support DLLs will also support Prode Properties. There are, however, some exceptions in passing strings (arrays of characters) since Microsoft utilizes proprietary data formats. Prode Properties includes Microsoft specific methods in addition to the standard methods supporting the ANSI C standard, Microsoft specific methods are compatible with almost all Microsoft applications.

Define models, compatibility with old versions

Prode Properties includes many methods for defining (via software) the thermodynamic models and the related options, see the paragraph “Methods to set / access different options” for additional information
Prode Properties allows to define via software the units of measurement, there are methods for defining the units and methods for retrieving codes and strings, see paragraph “Methods for accessing / defining the units of measurement” and the samples provided with the package for additional information, in Prode Properties to reference a unit must use a numeric code.

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>UNIT</th>
<th>CODE</th>
<th>DEFAULT UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure (abs)</td>
<td>CONV_P</td>
<td>15</td>
<td>“Pa.a”</td>
</tr>
<tr>
<td>Pressure (rel)</td>
<td>CONV_DP</td>
<td>16</td>
<td>“Pa”</td>
</tr>
<tr>
<td>Temperature (abs)</td>
<td>CONV_T</td>
<td>17</td>
<td>“K”</td>
</tr>
<tr>
<td>Temperature (rel)</td>
<td>CONV_DT</td>
<td>18</td>
<td>“K”</td>
</tr>
<tr>
<td>Calorific Value (weight)</td>
<td>CONV_HM</td>
<td>19</td>
<td>“Kj/Kg”</td>
</tr>
<tr>
<td>Calorific Value (molar)</td>
<td>CONV_HMM</td>
<td>20</td>
<td>“Kj/Kmol”</td>
</tr>
<tr>
<td>Power</td>
<td>CONV_HS</td>
<td>21</td>
<td>“KW”</td>
</tr>
<tr>
<td>Entropy (Streams)</td>
<td>CONV_SS</td>
<td>22</td>
<td>“KJ/(K*s)”</td>
</tr>
<tr>
<td>Heat Capacity (weight)</td>
<td>CONV_CP</td>
<td>23</td>
<td>“kJ/(kg*K)”</td>
</tr>
<tr>
<td>Heat Capacity (molar)</td>
<td>CONV_CPM</td>
<td>24</td>
<td>“kJ/(kmol*K)”</td>
</tr>
<tr>
<td>Flow (mass basis)</td>
<td>CONV_W</td>
<td>25</td>
<td>“Kg/s”</td>
</tr>
<tr>
<td>Flow (gas, mass basis)</td>
<td>CONV_WG</td>
<td>26</td>
<td>“Kg/s”</td>
</tr>
<tr>
<td>Density (weight)</td>
<td>CONV_D</td>
<td>27</td>
<td>“Kg/m³”</td>
</tr>
<tr>
<td>Density (molar)</td>
<td>CONV_DM</td>
<td>28</td>
<td>“Kmol/m³”</td>
</tr>
<tr>
<td>Specific Volume (weight)</td>
<td>CONV_SV</td>
<td>29</td>
<td>“m³/Kg”</td>
</tr>
<tr>
<td>Specific Volume (molar)</td>
<td>CONV_SVM</td>
<td>30</td>
<td>“m³/Kmol”</td>
</tr>
<tr>
<td>Thermal Conductivity</td>
<td>CONV_TC</td>
<td>31</td>
<td>“W/(m*K)”</td>
</tr>
<tr>
<td>Viscosity (dynamic)</td>
<td>CONV_V</td>
<td>32</td>
<td>“Pa*s”</td>
</tr>
<tr>
<td>Surface Tension</td>
<td>CONV_ST</td>
<td>33</td>
<td>“N/m”</td>
</tr>
<tr>
<td>Lenght</td>
<td>CONV_L</td>
<td>34</td>
<td>“m”</td>
</tr>
<tr>
<td>Area</td>
<td>CONV_A</td>
<td>35</td>
<td>“m²”</td>
</tr>
<tr>
<td>Volume</td>
<td>CONV_VOL</td>
<td>36</td>
<td>“m³”</td>
</tr>
<tr>
<td>Mass</td>
<td>CONV_M</td>
<td>37</td>
<td>“Kg”</td>
</tr>
<tr>
<td>Velocity</td>
<td>CONV_VL</td>
<td>38</td>
<td>“m/s”</td>
</tr>
<tr>
<td>Acceleration</td>
<td>CONV_ACC</td>
<td>39</td>
<td>“m/s²”</td>
</tr>
<tr>
<td>Force</td>
<td>CONV_FOR</td>
<td>40</td>
<td>“N”</td>
</tr>
<tr>
<td>Time</td>
<td>CONV_TM</td>
<td>41</td>
<td>“s”</td>
</tr>
<tr>
<td>Heat Flux</td>
<td>CONV_HF</td>
<td>42</td>
<td>“KW/m²”</td>
</tr>
<tr>
<td>Thermal Resistance</td>
<td>CONV_TR</td>
<td>43</td>
<td>“K*m²/KW”</td>
</tr>
<tr>
<td>Heat Transfer Coefficient</td>
<td>CONV_HTC</td>
<td>44</td>
<td>“KW/(m²*C)”</td>
</tr>
<tr>
<td>Flow (volume basis)</td>
<td>CONV_VW</td>
<td>45</td>
<td>“m³/s”</td>
</tr>
<tr>
<td>Viscosity (kinematic)</td>
<td>CONV.VK</td>
<td>46</td>
<td>“m²/s”</td>
</tr>
<tr>
<td>Energy</td>
<td>CONV_EN</td>
<td>47</td>
<td>“kJ”</td>
</tr>
<tr>
<td>Dipole moment</td>
<td>CONV_EDM</td>
<td>48</td>
<td>“c-m”</td>
</tr>
<tr>
<td>Solubility parameter</td>
<td>CONV_SP</td>
<td>49</td>
<td>“(J/m³)^1/2”</td>
</tr>
<tr>
<td>Flow Coefficient</td>
<td>CONV_CV</td>
<td>50</td>
<td>“Cv”</td>
</tr>
<tr>
<td>Compressibility coefficient</td>
<td>CONV_CC</td>
<td>51</td>
<td>“1/Pa”</td>
</tr>
<tr>
<td>Joule Thomson coefficient</td>
<td>CONV_JTC</td>
<td>52</td>
<td>“K/Pa”</td>
</tr>
<tr>
<td>Flow (molar basis)</td>
<td>CONV_WM</td>
<td>53</td>
<td>“Kmol/s”</td>
</tr>
<tr>
<td>Volume expansivity</td>
<td>CONV_VE</td>
<td>54</td>
<td>“1/K”</td>
</tr>
</tbody>
</table>
Introducing Prode Properties library methods

Prode Properties library includes a range of methods to deal with problems in chemical engineering and to achieve tight control over the calculations.

A non-inclusive list would include
- Thermodynamic calcs (flash operations, enthalpy, entropy, volume, energy, unit operations)
- Streams data access and calcs (set and retrieve operating conditions, critical and transport properties calcs)
- Chemicals library access (retrieve data from chemicals file)
- Error messages (management of errors messages)

Methods for thermodynamic calc’s

Prode Properties includes a complete set of methods for solving all the standard flash operations with specified final temperature or pressure and entropy or enthalpy or volume or energy basis, phase fraction with temperature or pressure basis plus mixers, dividers, gas,liquid phase separation operations etc.

integer result = setOp(integer stream, double t, double p)
Given a stream, operating pressure and temperature, performs an isothermal flash and sets operating conditions.

integer result = setSOp(integer stream)
Given a stream performs an isothermal flash at (user defined) standard conditions.

double t = PfPF(integer stream, double p, double pf, int state, int n)
Given a stream, the pressure, phase fraction (range 0-1), state (gas, liquid, solid) and position n calculates and returns the nth (n : 1-5) equilibrium temperature along the specified phase fraction line

double p = PfTF(integer stream, double t, double pf, int state, int n)
Given a stream, the temperature, phase fraction (range 0-1), state (gas, liquid, solid) and position n calculates and returns the nth (n : 1-5) equilibrium pressure along the specified phase fraction line

double t = LfPF(integer stream, double p, double lf)
Given a stream, the pressure and Liquid fraction (range 0-1) calculates and returns the first equilibrium temperature along the specified phase fraction line

double p = LfTF(integer stream, double t, double lf)
Given a stream, the temperature and Liquid fraction (range 0-1) calculates and returns the first equilibrium pressure along the specified phase fraction line

double t = HPF(integer stream, double p, double h, double et)
Given a stream, final pressure, the required (final) enthalpy (see the method StrH() for the definition) and a estimated value for final temperature (or 0 for automatic estimate), method solves the flash operation (enthalpy basis) and returns final temperature

double p = HTF(integer stream, double t, double h, double ep)
Given a stream, final temperature, the required (final) enthalpy (see the method StrH() for the definition) and a estimated value for final pressure (or 0 for automatic estimate), method solves the flash operation (enthalpy basis) and returns final pressure

double t = SPF(integer stream, double p, double s, double et)
Given a stream, final pressure, the required (final) entropy (see the method StrS() for the definition) and a estimated value for final temperature (or 0 for automatic estimate), method solves the flash operation (entropy basis) and returns final temperature.

double p = STF(integer stream, double t, double s, double ep)
Given a stream, final temperature, the required (final) entropy (see the method StrS() for the definition) and a estimated value for final pressure (or 0 for automatic estimate), method solves the flash operation (entropy basis) and returns final pressure.

double t = VPF(integer stream, double p, double v, double et)
Given a stream, final pressure, the required specific volume (see the method StrV() for the definition) and a estimated value for final temperature (or 0 for automatic estimate), method solves the flash operation (volume basis) and returns final temperature.

double p = VTF(integer stream, double t, double v, double ep)
Given a stream, final temperature, the required specific volume (see the method StrV() for the definition) and a estimated value for final pressure (or 0 for automatic estimate), method solves the flash operation (volume basis) and returns final pressure.

integer result = HVF(integer stream, double h, double v, double et, double ep)
Given a stream, the required (final) enthalpy (see the method StrH() for the definition) the required (final) specific volume (see the method StrV() for the definition) and estimated values for final temperature and pressure (or 0 for automatic estimate), method solves the flash operation.

integer result = SVF(integer stream, double s, double v, double et, double ep)
Given a stream, the required (final) entropy (see the method StrS() for the definition) the required specific volume (see the method StrV() for the definition) and estimated values for final temperature and pressure (or 0 for automatic estimate), method solves the flash operation.
Methods for stream’s data access

Prode Properties includes a set of functions for accessing stream parameters and calculating transport properties. Note that when calculating transport properties the program performs a VLE flash and returns 0 (zero value) when no associated liquid or gas phase is found.

```c
integer res = isSDef(integer stream)
given a stream returns TRUE (integer = 1) if stream has been defined, otherwise returns FALSE (0)

do double t = getT(integer stream)
given a stream returns stream’s operating temperature

do double p = getP(integer stream)
given a stream returns stream’s operating pressure

integer nr = getPNr()
returns the maximum number of phases that procedure can detect

integer type = StrPt(integer stream, int phase)
given a stream and position in range 1- getPNr() returns the phase type (vapor, liquid, solid)

char *description = StrPts(integer stream, int phase)
given a stream and position in range 1- getPNr() returns a ANSI C string with the description (vapor, liquid, solid...)

int description MStrPts(integer stream, int phase, char *s, integer slm)
given a stream and position in range 1- getPNr() fills string s with the description (vapor, liquid, solid...) (eventually truncated to slm maximum length), this is the Microsoft Excel specific method

do double lf = StrLf(integer stream)
given a stream returns the total liquid fraction (molar basis) in stream

do double pf = StrPf(integer stream, integer phase)
given a stream and phase position in range 1- getPNr() returns the phase fraction

do double w = getW(integer stream, integer phase, integer pos.)
given a stream, the phase position and component’s position (in component’s list) returns the component molar fraction in that phase

do double Zi= getZ(integer stream, integer pos.)
given a stream and component’s position (in component’s list) returns the comp’s Z (weight percentage, molar basis)

integer res = putZ(integer stream, integer pos., double Zi)
given a stream, comp’s position and Z , sets the comp’s pos. in Z vector (composition, molar basis) for that stream

integer nr = getCNr(integer stream)
given a stream returns the number of components defined in that stream

integer nr = getMCNr()
returns the maximum number of components in a stream

double zv = StrZv(integer stream)
given a stream returns the relevant compressibility factor (gas phase)

do double mw = StrMw(integer stream)
given a stream returns the averaged molecular weight (all phases)

double v = StrV(integer stream)
given a stream returns the specific volume as sum of specific volumes of all phases

double mw = StrGMw(integer stream)
given a stream returns the averaged molecular weight (gas phase)

double mw = StrLMw(integer stream)
given a stream returns the averaged molecular weight (liquid phase)

do double h = StrH(integer stream)
given a stream returns the total (stream) enthalpy (gas + liquid + solid phases)
```
double h = StrGH(integer stream)
given a stream returns the total (stream) enthalpy (gas phase)

double h = StrSGH(integer stream)
given a stream returns the specific (unit weight) enthalpy (gas phase)

double h = StrLH(integer stream)
given a stream returns the total (stream) enthalpy (liquid phase)

double h = StrSLH(integer stream)
given a stream returns the specific (unit weight) enthalpy (liquid phase)

double h = StrSH(integer stream)
given a stream returns the total (stream) enthalpy (solid phase)

double h = StrSSH(integer stream)
given a stream returns the specific (unit weight) enthalpy (solid phase)

double cp = StrGICp(integer stream)
given a stream returns the ideal gas heat capacity

double cp = StrGCp(integer stream)
given a stream returns the specific heat capacity (constant pressure, gas phase)

double cv = StrGCv(integer stream)
given a stream returns the specific heat capacity (constant volume, gas phase)

double cp = StrLCp(integer stream)
given a stream returns the specific heat capacity (constant pressure, liquid phase)

double cv = StrLCv(integer stream)
given a stream returns the specific heat capacity (constant volume, liquid phase)

double cp = StrSCp(integer stream)
given a stream returns the specific heat capacity (constant pressure, solid phase)

double ss = StrMSS(integer stream)
given a stream returns the speed of sound (gas, liquid) as calculated with HEM model for mixed phases

double ss = StrGSS(integer stream)
given a stream returns the speed of sound in gas phase

double ss = StrLSS(integer stream)
given a stream returns the speed of sound in liquid phase

double jt = StrGJT(integer stream)
given a stream returns the Joule Thomson coefficient in gas phase

double jt = StrLJT(integer stream)
given a stream returns the Joule Thomson coefficient in liquid phase

double ic = StrGIC(integer stream)
given a stream returns the isothermal compressibility coefficient \(- (1 / V) * dV / dP\) in gas phase

double ic = StrLIC(integer stream)
given a stream returns the isothermal compressibility coefficient \(- (1 / V) * dV / dP\) in liquid phase

double v = StrGVE(integer stream)
given a stream returns the volumetric expansivity coefficient \((1 / V) * dV / dT\) in gas phase

double ic = StrLVE(integer stream)
given a stream returns the volumetric expansivity coefficient \((1 / V) * dV / dT\) in liquid phase
double s = StrGS(integer stream)
given a stream returns the total (stream) entropy (gas phase)

double s = StrSGS(integer stream)
given a stream returns the specific (unit weight) entropy (gas phase)

double s = StrLS(integer stream)
given a stream returns the total (stream) entropy (liquid phase)

double s = StrSS(integer stream)
given a stream returns the total (stream) entropy (solid phase)

double s = StrSLS(integer stream)
given a stream returns the specific (unit weight) entropy (liquid phase)

double s = StrSSS(integer stream)
given a stream returns the specific (unit weight) entropy (solid phase)

double s = StrS(integer stream)
given a stream returns the total (stream) entropy (gas + liquid + solid phases)

integer res = setWm(integer stream, double W)
given a stream and flow (mass basis), sets the flow

double w = getWm(integer stream)
given a stream returns the flow specified for that stream.

double hc = StrHC(integer stream)
given a stream returns the calculated net heat of combustion (gas phase).

double fl = StrFML(integer stream)
given a stream returns the calculated flammability lean limit (gas phase).

double fl = StrFMH(integer stream)
given a stream returns the calculated flammability rich limit (gas phase).

double d = StrLD(integer stream)
given a stream returns the calculated liquid density (at operating conditions).

double d = StrGD(integer stream)
given a stream returns the calculated gas density (at operating conditions).

double tc = StrLC(integer stream)
given a stream returns the calculated liquid thermal conductivity (at operating conditions).

double tc = StrGC(integer stream)
given a stream returns the calculated gas thermal conductivity (at operating conditions).

double v = StrLV(integer stream)
given a stream returns the calculated liquid viscosity (at operating conditions).

double v = StrGV(stream)
given a stream returns the calculated gas viscosity (at operating conditions).

double st = StrST(integer stream)
given a stream returns the calculated surface tension (at operating conditions).

Integer cpnr = StrCPnr(integer stream)
given a stream returns the number of critical points detected and calculated, to get a critical point use the methods StrPc() and StrTc() setting value of pos in the range 1-cpnr

double p = StrPc(integer stream, Integer pos)
given a stream and the critical point position in the list (see method StrCPnr()) returns the critical pressure

double t = StrTc(integer stream, Integer pos)
given a stream and the critical point position in the list (see method StrCPnr()) returns the critical temperature.
double p = StrCBp(integer stream)
given a stream returns the cricodenBar pressure.

double t = StrCBt(integer stream)
given a stream returns the cricodenBar temperature.

double p = StrCTp(integer stream)
given a stream returns the cricodenTherm pressure.

double t = StrCTt(integer stream)
given a stream returns the cricodenTherm temperature.

double ac = StrAc(integer stream)
given a stream returns the acentric factor (mole fraction average).

double p = StrRVP(integer stream, integer mode)
given a stream returns the Reid vapor pressure
mode = 1 simulation of D6377 procedure (liquid not saturated with air)
mode = 2 simulation of D323 procedure (liquid saturated with air)

double fp = StrFLP(integer stream)
given a stream returns the Flash point (for pure fluids the method returns the value stored in databank while for mixtures the flash point is calculated by a iterative procedure where VLE is solved according the selected models for stream)
Methods for stream's definition

Prode Properties includes a set of functions to define a stream by program (as alternative to utilize the Properties Editor)

- to create a NEW list of components, call initS() and define the list of components with putCC()
- define the mole fraction of each component with putZ()
- call setS() to define the stream
- call setW() to define the flow
- utilize the methods described in paragraph “Methods to define thermodynamic models” to define the models
- call loadSB() to load the BIPs from database or define specific BIPs with methods PutCi(), PutCj(), PutMB(), PutBIP()

integer res = initS (integer stream)
given a stream initializes all data, call this method before to create a new list of components.

integer res = putCC (integer stream, integer pos, integer compcode)
given a stream, component's position (in component's list) and component code sets the code in component's list.

integer res = putZ(integer stream, integer pos., double Zi)
given a stream, comp's position and Z, sets the comp's pos. in Z vector (composition, molar basis) for that stream

integer res = setS(integer stream)
given a stream performs a sequence of validating operations on data. This method must be called after to have restored stream's data from archives (files etc.)

Methods to define a initial condition for a stream

integer res = loadSB(integer stream, integer btype)
given a stream loads all BIP available in database. This method must be called after the stream has been defined since it requires the list of components. Codes for btype are 0 for VLE, 1 for LLE, 2 for SLE, 3 for Hydrates

double Zi= getZ(integer stream, integer pos.)
given a stream and component's position (in component's list) returns the comp's Z (molar fraction)

integer cc = getCC(integer stream, integer pos)
given a stream and component's position (in component's list) returns the component code (a integer that identifies the component in chemical's file).

integer nr = getMBPNr()
returns the maximum number of (interaction coefficients) binary pairs in a stream

int ci = getCi(integer stream, integer pos)
given a stream and position (in interaction's coeff. list) returns the first component reference (a integer that identifies the component in component's list).

integer res = PutCi (integer stream, integer pos, integer ci)
given a stream, position (in interaction coefficients list) and first component reference sets the component's reference in interaction coefficient's list.

int cj = getCj(integer stream, integer pos)
given a stream and position (in interaction’s coeff. list) returns the second component reference (an integer that identifies the component in component's list).

integer res = PutCj (integer stream, integer pos, integer cj)
given a stream, position (in interaction coefficients list) and second component reference sets the component's reference in interaction coefficient's list.

int model = getMB(integer stream, integer pos)
given a stream and position (in interaction's coeff. list) returns the related model (an integer that identifies the model).

integer res = PutMB(integer stream, integer pos, integer model)
given a stream, position (in interaction coefficients list) and a model identifier sets the model in interaction coefficient's list.

double BIP = getBIP(integer stream, integer pos, integer id)
given a stream, position (in binary coeff. list) and BIP identifier (0-max nr. of BiPs for that model) returns BIP.

integer res = PutBIP(integer stream, integer pos, integer id. double Kji)
given a stream, position (in binary coeff. list) BIP identifier (0-max nr. of BiPs for that model) and value stores BIP in that position of the list.
Methods to define stream’s operating conditions

Prode Properties includes a set of functions to define the initial (operating) condition of a stream, these can be utilized as an alternative to the standard initialization via setOp() method

- call rstValidSop()
- define composition in phase 1 with putW()
- define phase fraction with putPF()
- define phase type with putPT()
- set phase 1 as valid , setValidPhase()
- continue with another phase (2...n)
- define temperature with putT()
- define pressure with putP()
- set conditions as valid with setValidSop()

integer result = rstValidSop(integer stream)
Given a stream clears the compositions of different phases at operating conditions

integer result = setValidSop(integer stream)
Given a stream sets the compositions of different phases at operating conditions as valid.

integer result = setValidPhase(integer stream, integer phase)
Given a stream and phase sets the phase composition as valid.

integer result = putW(integer stream, integer phase, int compnr, double w)
Given a stream, phase, component number and component’s molar fraction in that phase stores the value

integer result = putPF(integer stream, integer phase, double fraction)
Given a stream, phase and phase fraction stores the phase .fraction value

integer result = putPT(integer stream, integer phase, int type)
Given a stream, phase and phase type (vapor, liquid, solid) stores the phase type

integer result = putT(integer stream, double t)
Given a stream and operating temperature stores the value

integer result = putP(integer stream, double p)
Given a stream and operating pressure stores the value

Copy of streams

to make a copy of a stream utilize the method

integer res = StrCopy(integer stream1, integer stream2)
Given two streams (stream1 and stream2) copies the stream 2 into stream 1
Methods for solving staged columns

Note: this method utilizes the standard ANSI C convention for exchanging parameters (see the samples provided with the software)

Properties includes a procedure for solving staged columns, the column is modeled with stgnr equilibrium stages, column
column may include a condenser and a reboiler, stage numbering is bottom up, the bottom stage (reboiler, if specified) is number one
and the top stage (condenser, if specified) is number stgnr.
There may be one or more feeds, a feed is modeled by entering liquid on the specified stage and vapor portion to the stage
above (with exception of top stage).
There may be one or more side streams.
Heat added/removed on each stage can be specified.
Efficiency parameter on each stage can be specified.

integer res = DCOL(int csep, int stgnr, int init, double *stgt,double *stgp,double *stgef,double *stgdH,
int prod_h, int btm_h,int fnr,int *fstr,int *fpos, int snr, int *sstr,int *spos,int *sft,
double *sflow, int vnr, double *vrv,int *vtype,int *ptype,int *piv,double *prv,
double*flows)

Parameters:
csep (int) column type: 1 VLE, 2 VLLE, 3 LLE (some features available in extended versions)
stgnr (int) number of stages
init (int) 0 for automatic initialization, 1 temperatures and flows are defined by user
stgt (double*) vector (stgnr) with stage temperatures
stgp (double*) vector (stgnr) with specified stage pressures
stgef (double*) vector (stgnr) with specified stage efficiency, permitted range 0,1-1
stgdH (double*) vector (stgnr) with specified dH (heat added, removed)
prod_h (int) stream for top product/distillate
btm_h (int) stream for bottom product
fnr (int) number of feeds
fstr (int*) vector (fnr) with the feeding streams
fpos (int*) vector (fnr) with feeds positions 1-stgnr
snr (int) number of side streams
sstr (int*) vector (snr) with the list of side streams
spos (int*) vector (snr) with side streams positions (1-stgnr)
sft (int*) vector (snr) with specified flow type (GAS_PHASE, LIQ_PHASE, see Codes used in Prode library)
sflow (double*) vector (snr) with the specified (on each side stream) side product to feed flow ratio
vnr (int) number of variables to solve
vtype (int*) vector (vnr) with type of variable (see below)
vrv (double*) vector (vnr) with calculated values for variable
ptype (int*) vector (pnr) with type of specification (see below)
piv (int*) vector (pnr) with integer values as the position of components in the list
prv (double*) vector (pnr) with values of the specifications to solve
flows (double*) vector with calculated values for vapor/liquid flows in all stages, dimension nrphases*nrc*stgnr

Codes for variables
reboiler 1
total condenser 2
partial condenser 3

Codes for specifications
reflux ratio 1
product to feed ratio (molar fract.) 2
bottom to feed ratio (molar fract.) 3
component (molar fract.) in top product 4
component (molar fract.) in bottom product 5
component recovery in top product 6
component recovery in bottom product 7

Notes:
When passing / returning parameters the first element in vectors is the element 0.
Main variables (1-vnr) are (when specified) reboiler and condenser (partial or total), each variable (of type defined in vtype)
requires a suitable specification (in ptype, piv, prv), usually for reboiler the specification is the product to feed ratio and for a
condenser the reflux ratio, but specifications based on component's fractions on top and bottom products are permitted, in
these cases specify in piv the position of selected component in the list and in prv the value of the fraction required.

73
Secondary variables are side streams (1-snr), each side stream (defined in sstr, spos) requires (in sflow) a specification for the side product to (total) feed flow ratio.
The column is modeled with thermodynamics and options defined for the first feed in the list.

**Initialization**
in most cases the procedure doesn't require to initialize values, when required set the variable init to 1 and define the proper initial values in vectors stgt and flows, note that in a sequence of similar operations (for example when controlling the operating point of a column) it may result useful to reintroduce the calculated values as starting point for the new calculus

**Examples**
Column with 8 stages, 1 feed (stage 4), pressure reboiler 12.5 Bar, pressure top 12 Bar, efficiency 1, $\Delta H = 0$

```
csep = 1
stgnr = 8
init = 0
stgp[0] = 12.5
......
stgp[7] = 12
stgef[0] = 1
......
stgef[7] = 1
stgdH[0] = 0
......
stgdH[7] = 0
prod_h = stream1
btm_h = stream2
fnr = 1
fstr = stream3
fpos = 4
```
variables : reboiler and total condenser
specifications : component 2 fraction in top product and bottom product to feed ratio

```
vnr = 2
vtype[0] = 1
ptype[0] = 4
piv[0] = 2
prv[0] = 0.96
vtype[1] = 2
ptype[1] = 3
piv[1] = 0
prv[1] = 0.4
```
variables : reboiler, partial condenser and 2 side streams (liquid and gas phases)
specifications : component 2 fraction in top product, bottom product to feed ratio, side streams flow to feed ratio

```
vnr = 2
vtype[0] = 1
ptype[0] = 4
piv[0] = 2
prv[0] = 0.96
vtype[1] = 3
ptype[1] = 3
piv[1] = 0
prv[1] = 0.4
```

```
snr = 2
sstr[0] = stream4
spos[0] = 4
sft[0] = LIQ_PHASE
sflow[0] = 0.12
sstr[1] = stream5
spos[1] = 7
sft[1] = GAS_PHASE
sflow[1] = 0.078
```
Note: this method utilizes the standard ANSI C convention for exchanging parameters. (see the samples provided with the software)

simulation of reactors

```c
int res = REACT(int streamIn, streamOut, int model, int NrReactions, double **Conv, double Pout, double dHeat)
```

Parameters:
- `streamIn` (int): inlet stream
- `streamOut` (int): outlet stream
- `model` (int): model for reactor (see below)
- `NrReactions` (int): number of reactions
- `Conv` (double**): matrix (NrComponents, NrReactions) to specify reactions
- `Pout` (double): output pressure
- `dHeat` (double): heat added, removed

**Codes for models**
- Gibbs: 1
- Equilibrium Reactor: 2
- Additional models on request
Methods for fluid flow problems

Simulation of single phase, two-phases, multiphase flow on circular pipes

```c
int res = PIPE(int stream, int model, double diam, double rough, double length, double dHeight, double dHeat)
```

**Parameters:**
- `stream` (int) inlet stream
- `model` (int) model for fluid flow and phase equilibria (see below)
- `diam` (double) pipe internal diameter
- `rough` (double) parameter defining relative pipe roughness
- `length` (double) length of this segment
- `dHeight` (double) height difference (inlet, outlet)
- `dHeat` (double) heat added, removed

**Codes for models**
- Beggs & Brill / Hazen-Williams / AGA 1
- Additional models on request

Methods for Hydrates phase equilibria

Methods for calculating hydrate formation pressure (or temperature)

```c
double p = HPFORM(int stream, double t, int method)
double t = HTFORM(int stream, double p, int method)
```

**Parameters:**
- `stream` (int) inlet stream
- `t` (double) operating temperature (or operating pressure)
- `method` (int)
  - 1 = include SI , SII , SH
  - 2 = SI
  - 3 = SII

Methods for solving a Polytropic operation

Polytropic stage (compression and expansion)

```c
double val = PSPF(int stream, double pout, int model, double param)
```

**Parameters:**
- `stream` (int) inlet stream
- `pout` (double) outlet pressure
- `model` (int) model, see below codes 1-4
- `param` (double) for model 1 and 3 specified polytropic efficiency (range 0-1)
  - for model 2 and 4 (measured) outlet temperature

The procedure can model compression and expansion units such as centrifugal compressors, expansion turbines etc. The procedure returns:
- calculated temperature options 1,3
- calculated efficiency options 2,4

**Models available (**)**

1. Given initial condition, pout and polytropic efficiency calculates outlet condition,
   - Method applicable to gas phase only

2. Given initial condition, pout and tout calculates polytropic efficiency,
   - Method applicable to gas phase only

3. Given initial condition, pout and polytropic efficiency calculates outlet condition
   - R.Paron "Polytropic solution with phase equilibria"
   - Method applicable to gas and mixed (gas + liquid) phases

4. Given initial condition, pout and tout calculates polytropic efficiency,
   - R.Paron "Polytropic solution with phase equilibria"
   - Method applicable to gas and mixed (gas + liquid) phases

(**) Additional models available from Prode
This unit models a relief valve with different methods.

\[
\text{double area} = \text{ISPF(int stream, double pout, int model, double *param)}
\]

Parameters:
- \text{stream (int)}: inlet stream
- \text{pout (double)}: outlet pressure
- \text{model (int)}: model, see below codes 1-4
- \text{param(double)}: correction parameter, see below the range of values

The procedure models a relief valve at specified operating conditions and returns the calculated area.

Models available (**):
1. \text{HEM: Homogeneous Equilibrium (Solution of Mass Flux integral)}
2. \text{HNE: Homogeneous Non-equilibrium (HEM with Boling Delay and Gas-Liquid Slip Contributes)}
3. \text{HNE-DS: Homogeneous Non-equilibrium}
4. \text{NHNE: Non-homogeneous Non-equilibrium}

Model recommended range of values for correction parameter:
- \text{HEM: not required}
- \text{HNE: 0.7-0.8 for safety valves}
- \text{HNE-DS: see the paper}
- \text{NHNE: 0.7-0.8 for safety valves}

(**) Additional models available from Prode
Methods for calculating equilibrium lines in phase diagrams

Note: these methods utilize the standard ANSI C convention for exchanging parameters, the distribution includes samples to show how to utilize these methods in different languages.

Prode Properties includes methods for calculating different types of phase diagrams:
- vapor-liquid
- vapor-liquid-liquid
- vapor-liquid-solid (**)

(**) feature available in extended versions

Typical application:
- Call PELDr() to calculate the phase diagram and obtain the number of lines available.
- On each line call PELP(), PELT(), PELine() to obtain the data for the different lines.
- If required call PFLine() to calculate a line with specified phase fraction and state.

**integer lnr = PELDr(integer stream)**
Given a stream, calculates the phase diagram and returns the number of equilibrium lines available.

**integer lnr = PELT(integer stream, integer line)**
Given a stream and the line, returns the line type (see below).

1 = bubble line
2 = dew line
3 = three phase line

**integer lnr = PELP(integer stream, integer line)**
Given a stream and the line, returns the line property (see below).

1 = vapor-liquid
2 = vapor-liquid-liquid
3 = vapor-solid
4 = liquid-solid

**integer nrpt = PELine(integer stream, integer line, double *P, double *T, int maxpt)**
Given a stream, the line and two arrays (0-maxpt elements) the procedure returns nrpt < maxpt equilibrium points in specified line.

**integer nrpt = PVLine(integer stream, integer line, double *P, double *T, double *H, double *S, double *V, int maxpt)**
Given a stream, the line and five arrays (0-maxpt elements) the procedure returns nrpt < maxpt equilibrium points in specified line,
in additions to t,p values this method returns enthalpy, entropy and volume values calculated at equilibrium points.

This method allows to calculate a line with specified phase fraction in specified state (gas, liquid, solid).

**integer nrpt = PFLine(integer stream, integer line, double pf, double *P, double *T, int maxpt)**
Given a stream, the line, a specified phase fraction and two arrays (0-maxpt elements) the procedure returns nrpt < maxpt equilibrium points in specified phase fraction line.
Methods for direct access to properties (F,H,S,V) and derivatives (T,P,W)

Prode Properties includes methods for fast access to the procedures for calculating thermodynamic properties, to
access these methods create one or more processes with method DPinit() passing a integer as process code
(permitted range of values 1..5) to identify each process and a stream (caution: must define the stream, i.e. the list
of components and molar fractions, before to call DPinit), then call in sequence the methods for calculating ther-
mosdynamic properties passing as first parameter the process code, the methods will return the properties calcul-
fled for the stream associated with that specific process.
Note: Base version allows to define up to 5 iindependent processes

eample of application
DPinit(1,stream);
StrHv(1,0,t,p,X,&HL);
StrHv(1,1,t,p,Y,&HV);

integer res = DPinit(integer process,integer stream)
Given a process (code 1-5) and a stream the method loads all data

integer res = StrFv(integer process,integer state,double t ,double p, double *w,double *fg)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the vector of fugacities
(Pa)

integer res = StrFv(integer process,integer state,double t ,double p, double *w,double *fg, double *dfgt,
double *dfgp, double **dfgw)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the vector of fugacities
(Pa) and related derivatives vs. temperature (K), pressure (Pa), composition (note: derivatives vs. composition as matrix [n][m])

integer res = StrFvd(integer process,integer state,double t ,double p, double *w,double *fg, double *dfgt,
double *dfgp, double **dfgw)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the vector of fugacities
(Pa) and related derivatives vs. temperature (K), pressure (Pa), composition (note: derivatives vs. composition as vector [n*m])

integer res = StrHv(integer process,integer state,double t ,double p, double *w,double *H)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the molar enthalpy (Kj/Kmol)

integer res = StrHvd(integer process,integer state,double t ,double p, double *w,double *H, double *dHt,
double *dHp, double *dHw)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the molar enthalpy (Kj/Kmol) and related derivatives vs. temperature, pressure, composition

integer res = StrSv(integer process,integer state,double t ,double p, double *w,double *S)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the molar entropy (Kj/Kmol-K)

integer res = StrSvd(integer process,integer state,double t ,double p, double *w,double *S, double *dSt,
double *dSp, double *dSw)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the molar entropy (Kj/Kmol-K) and related derivatives vs. temperature, pressure, composition

integer res = StrVv(integer process,integer state,double t ,double p, double *w,double *V)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the molar volume (M3/Kmol)

integer res = StrVvd(integer process,integer state,double t ,double p, double *w,double *V, double *dVt,
double *dVp, double *dVw)
Given a predefined stream the required state (vapor, liquid, solid) and operating conditions returns the molar volume (M3/Kmol) and related derivatives vs. temperature, pressure, compositionMethods for stream’ s data access
Extended methods for accessing stream's properties

These functions (which are otherways equivalent to standard methods) permit in addition to set the operating conditions at which the required property must be evaluated. This may result useful in many cases, for example when utilizing Prode Properties methods as macros from Excel cells. Caution : the isothermal flash will reset any previous settings and you should use with care these methods in sequential calculus.

```c
double mw = EStrGMw(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the molecular weight for gas phase

double mw = EStrLMw(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the molecular weight for liquid phase

double lf = EStrLf(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns liquid fraction (molar basis) in stream

double pf = EStrPf(integer stream, integer state, double t, double p)
given a stream, state (gas, liquid, solid) pressure and temperature performs an isothermal flash and returns the phase fraction (molar basis) in specified state

double zv = EStrZv(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the relevant compressibility factor (gas phase)

double h = EStrH(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the enthalpy (gas + liquid phase)

double v = EStrV(integer stream, double t, double p)
given a stream, pressure and temperature performs an isothermal flash and returns the specific volume as sum of specific volumes of all phases

double cp = EStrGCp(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the specific heat capacity (constant pressure, gas phase)

double cv = EStrGCv(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the specific heat capacity (constant volume, gas phase)

double cp = EStrLCp(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the specific heat capacity (constant pressure, liquid phase)

double cv = EStrLCv(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the specific heat capacity (constant volume, liquid phase)

double c = EStrGIC(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the isothermal compressibility in gas phase

double c = EStrLIC(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the isothermal compressibility in liquid phase

double ss = StrMSS(integer stream, double t, double p)
given the stream pressure and temperature performs an isothermal flash and returns the speed of sound (gas, liquid) as calculated with HEM model for mixed phases

double ss = EStrGSS(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the speed of sound in gas phase

double ss = EStrLSS(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the speed of sound in liquid phase

double jt = EStrGJT(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the Joule Thomson coefficient for gas phase
```
double jt = EStrLJT(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the Joule Thomson coefficient for liquid phase

double ic = EStrGIC(integer stream double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the isothermal compressibility coefficient \(- (1 / V) * dV / dP\) in gas phase

double ic = EStrLIC(integer stream double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the isothermal compressibility coefficient \(- (1 / V) * dV / dP\) in liquid phase

double v = EStrGVE(integer stream double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the volumetric expansivity coefficient \(- (1 / V) * dV / dT\) in gas phase

double v = EStrLVE(integer stream double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the volumetric expansivity coefficient \(- (1 / V) * dV / dT\) in liquid phase

double hc = EStrHC(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the net heat of combustion (gas phase).

double fl = EStrFML(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the flammability lean limit (gas phase).

double fl = EStrFMH(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the flammability rich limit (gas phase).

double s = EStrS(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the relative entropy (gas + liquid phase)

double d = EStrLD(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the calculated liquid density (at operating conditions).

double d = EStrGD(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the calculated gas density (at operating conditions).

double tc = EStrLC(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the calculated liquid thermal conductivity (at operating conditions).

double tc = EStrGC(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the calculated gas thermal conductivity (at operating conditions).

double v = EStrLV(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the calculated liquid viscosity (at operating conditions).

double v = EStrGV(stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the calculated gas viscosity (at operating conditions).

double st = EStrST(integer stream, double t, double p)
given the stream, pressure and temperature performs an isothermal flash and returns the calculated surface tension (at operating conditions).
Methods for chemical's file access

Prode Properties includes a set of functions for chemical data file access. Components are referenced via a component code which is an integer with value in the range 1 to getFCNR()

```
Integer nr = getFCNr()
returns the number of components in Chemical's File

int str = MCompF(integer code, char *s, integer slm )
given the component code fills string s with the relevant component formula (eventually truncated to slm maximum length), this is the Microsoft Excel specific method

char *str = CompF(integer code)
given the component code returns the relevant component formula (eventually truncated to string maximum length), this is the ANSI C compatible method

int str = MCompN(integer code, char *s, integer slm )
given the component code fills string s with the relevant component name (eventually truncated to slm maximum length), this is the Microsoft specific method

char *str = CompN(integer code)
given the component code returns the relevant component name (eventually truncated to string maximum length), this is the ANSI C compatible method

int id = CompID(integer code)
given the component code returns component’s ID (it’s the CAS number)

int cc = CompCID(integer id)
given the component ID returns the component’s code

double mw = CompMw(integer code)
given the component code returns the relevant molecular weight

double tc = CompTc(integer code)
given the component code returns the relevant critical temperature

double ac = CompAc(integer code)
given the component code returns the relevant acentric factor

double vc = CompVc(integer code)
given the component code returns the relevant critical volume

double pc = CompPc(integer code)
given the component code returns the relevant critical pressure

double dm = CompDm(integer code)
given the component code returns the dipole moment

double rg = CompRg(integer code)
given the component code returns the radius of gyration

double sol = CompSol(integer code)
given the component code returns the solubility parameter

double hf = CompHf(integer code)
given the component code returns the std. enthalpy of formation

double gf = CompGf(integer code)
given the component code returns the Gibbs energy of formation

double sf = CompSf(integer code)
given the component code returns the enthalpy of fusion

double nb = CompNb(integer code)
given the component code returns the normal boiling point
```
double mp = CompMp(integer code)  
given the component code returns the melting point

double p = CompVP(integer code, double t)  
given the component code and a temperature, returns the calculated saturation pressure (calculated via Chemical’s file temperature dependent correlation)

double h = CompHG(integer code, double t0, double t1)  
given the component code, initial and final temperatures for integration, returns the calculated ideal gas enthalpy (calculated via Chemical’s file temperature dependent correlation)

double s = CompSG(integer code, double t0, double t1)  
given the component code, initial and final temperatures for integration, returns the calculated ideal gas entropy (calculated via Chemical’s file temperature dependent correlation)

double h = CompHL(integer code, double t0, double t1)  
given the component code, initial and final temperatures for integration, returns the calculated ideal liquid enthalpy (calculated via Chemical’s file temperature dependent correlation)

double s = CompSL(integer code, double t0, double t1)  
given the component code, initial and final temperatures for integration, returns the calculated ideal liquid entropy (calculated via Chemical’s file temperature dependent correlation)

double h = CompHS(integer code, double t0, double t1)  
given the component code, initial and final temperatures for integration, returns the calculated ideal solid enthalpy (calculated via Chemical’s file temperature dependent correlation)

double s = CompSS(integer code, double t0, double t1)  
given the component code, initial and final temperatures for integration, returns the calculated ideal solid entropy (calculated via Chemical’s file temperature dependent correlation)

double h = CompHV(integer code, double t)  
given the component code and a temperature, returns the calculated latent heat (calculated via Chemical’s file temperature dependent correlation)

double v = CompLV(integer code, double t)  
given the component code and a temperature, returns the calculated liquid viscosity (calculated via Chemical’s file temperature dependent correlation)

double v = CompGV(integer code, double t)  
given the component code and a temperature, returns the calculated gas viscosity (calculated via Chemical’s file temperature dependent correlation)

double d = CompLD(integer code, double t)  
given the component code and a temperature, returns the calculated liquid density (calculated via Chemical’s file temperature dependent correlation)

double tc = CompLC(integer code, double t)  
given the component code and a temperature, returns the calculated liquid (thermal) conductivity (calculated via Chemical’s file temperature dependent correlation)

double tc = CompGC(integer code, double t)  
given the component code and a temperature, returns the calculated gas (thermal) conductivity (calculated via Chemical’s file temperature dependent correlation)

double st = CompST (integer code, double t)  
given the component code and a temperature, returns the calculated surface tension (calculated via Chemical’s file temperature dependent correlation)

double d = CompSD(integer code, double t)  
given the component code and a temperature, returns the calculated solid density (calculated via Chemical’s file temperature dependent correlation)

double tc = CompSC(integer code, double t)  
given the component code and a temperature, returns the calculated solid (thermal) conductivity (calculated via Chemical’s file temperature dependent correlation)
Methods to set / access different options

To set / access the different options available in Prode Properties the library includes several methods:

- `getKO()`, `putKO()` allow to access or define all the settings as a 32 bit integer, each bit in the integer represents a different option,
- `getKS()`, `putKS()` allow to access or define each option, see below a short list of options available

\[
\text{int value} = \text{getKO} (\text{integer stream})
\]

given a stream returns a code (integer) with the options

\[
\text{integer res} = \text{putKO} (\text{integer stream}, \text{integer value})
\]

given a stream define the options

\[
\text{int value} = \text{getKS} (\text{integer stream}, \text{integer option})
\]

given a stream and option (see below the codes) returns a boolean (0-1) with stored value

\[
\text{integer res} = \text{putKS} (\text{integer stream}, \text{integer option, integer value})
\]

given a stream and option define the option.

Codes used in Prode library

Table of codes to specify the different options

reference : methods `getKO()`, `setKO()` ...

Caution! The codes may change in different versions.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Decimal value</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>set multiphase vapor + liquid</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>set multiphase vapor + liquid + solid</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>set multiphase vapor + liquid + solid + hydrate</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>reduce the number of trial phases (in multiphase)</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>use iso compressibility coeff. to detect single phase state</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>evaluate stability of each phase in equilibrium</td>
</tr>
<tr>
<td>7</td>
<td>64</td>
<td>end specified phase fraction lines when crossing phase boundary lines</td>
</tr>
<tr>
<td>8</td>
<td>128</td>
<td>include all hydrate structures (also those not normally generated by formers)</td>
</tr>
</tbody>
</table>

to set one or more options call `setOM()` passing as value a integer with the sum (decimal values) of all required options.

Table of codes to specify the different states

reference : methods `setMP()`, `PTF()`, `PTF()`, `StrFv()`, `StrFvd()` ...

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Vapor phase</td>
</tr>
<tr>
<td>1</td>
<td>Liquid phase</td>
</tr>
<tr>
<td>2</td>
<td>Solid phase</td>
</tr>
<tr>
<td>3</td>
<td>Hydrate phase</td>
</tr>
</tbody>
</table>
Table of codes to specify the different models

reference: methods setMP(), getMP() ...

Some models may not be available and/or the numerical codes may change in different versions, contact Prode for details

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Regular</td>
<td>Regular</td>
</tr>
<tr>
<td>10</td>
<td>Wilson</td>
<td>Wilson</td>
</tr>
<tr>
<td>11</td>
<td>NRTL</td>
<td>NRTL</td>
</tr>
<tr>
<td>12</td>
<td>Soave-Redlich_Kwong Std. (VDW)</td>
<td>SRK(VDW)</td>
</tr>
<tr>
<td>30</td>
<td>Soave-Redlich_Kwong Ext. (VDW)</td>
<td>SRKXX(VDW)</td>
</tr>
<tr>
<td>40</td>
<td>Soave-Redlich_Kwong Ext. + NRTL (Modified Huron Vidal)</td>
<td>SRKXX-NRTL(P-HV)</td>
</tr>
<tr>
<td>41</td>
<td>Soave-Redlich_Kwong Ext. + NRTL (Modified LCVM)</td>
<td>SRKXX-NRTL(P-LCVM)</td>
</tr>
<tr>
<td>50</td>
<td>Peng Robinson Std. (VDW)</td>
<td>PR(VDW)</td>
</tr>
<tr>
<td>51</td>
<td>Peng Robinson Ext. (VDW)</td>
<td>PRX(VDW)</td>
</tr>
<tr>
<td>55</td>
<td>Peng Robinson Ext. + Wilson (Wong Sandler)</td>
<td>PRX-Wilson(WS)</td>
</tr>
<tr>
<td>56</td>
<td>Peng Robinson Ext. + NRTL (Wong Sandler)</td>
<td>PRX-NRTL(WS)</td>
</tr>
<tr>
<td>57</td>
<td>Peng Robinson Ext. + NRTL (Modified Huron Vidal)</td>
<td>PRX-NRTL(P-HV)</td>
</tr>
<tr>
<td>58</td>
<td>Peng Robinson Ext. + NRTL (Modified LCVM)</td>
<td>PRX-NRTL(P-LCVM)</td>
</tr>
<tr>
<td>59</td>
<td>Peng Robinson Ext. + NRTL (Modified MHV2)</td>
<td>PRX-NRTL(P-MHV2)</td>
</tr>
<tr>
<td>60</td>
<td>Peng Robinson Ext. + Wilson (MHV2)</td>
<td>PRX-Wilson(MHV2)</td>
</tr>
<tr>
<td>61</td>
<td>Peng Robinson Ext. + UNIQUAC (MHV2)</td>
<td>PRX-UNIQUAC(MHV2)</td>
</tr>
<tr>
<td>62</td>
<td>Peng Robinson Ext. + NRTL (MHV2)</td>
<td>PRX-NRTL(MHV2)</td>
</tr>
<tr>
<td>63</td>
<td>Peng Robinson Ext. + Wilson (Modified Huron Vidal)</td>
<td>PRX-Wilson(P-HV)</td>
</tr>
<tr>
<td>64</td>
<td>Peng Robinson Ext. + Wilson (Modified LCVM)</td>
<td>PRX-Wilson(P-LCVM)</td>
</tr>
<tr>
<td>65</td>
<td>Peng Robinson Ext. + Wilson (Modified MHV2)</td>
<td>PRX-Wilson(P-MHV2)</td>
</tr>
<tr>
<td>66</td>
<td>Peng Robinson Ext. + NRTL (Modified MHV2)</td>
<td>PRX-NRTL(P-MHV2)</td>
</tr>
<tr>
<td>67</td>
<td>Peng Robinson Ext. + NRTL (Modified LCVM)</td>
<td>PRX-NRTL(P-LCVM)</td>
</tr>
<tr>
<td>70</td>
<td>Peng Robinson Ext. + NRTL (Modified LCVM)</td>
<td>PRX-NRTL(P-LCVM)</td>
</tr>
<tr>
<td>71</td>
<td>Peng Robinson Ext. + Wilson (Modified LCVM)</td>
<td>PRX-Wilson(P-LCVM)</td>
</tr>
<tr>
<td>72</td>
<td>Peng Robinson Ext. + UNIQUAC (Modified LCVM)</td>
<td>PRX-UNIQUAC(P-LCVM)</td>
</tr>
<tr>
<td>73</td>
<td>Peng Robinson Ext. + UNIFAC (Modified LCVM)</td>
<td>PRX-UNIFAC(WS)</td>
</tr>
<tr>
<td>80</td>
<td>Benedict-Webb-Rubin (modified)</td>
<td>BWR</td>
</tr>
<tr>
<td>81</td>
<td>Benedict-Webb-Rubin-Starling</td>
<td>BWRS</td>
</tr>
<tr>
<td>90</td>
<td>Lee Kesler</td>
<td>LK</td>
</tr>
<tr>
<td>91</td>
<td>Lee Kesler Ploecker</td>
<td>LKP</td>
</tr>
<tr>
<td>100</td>
<td>P-SAFT</td>
<td>PSAFT</td>
</tr>
<tr>
<td>110</td>
<td>Soave-Redlich_Kwong Ext. - CPA</td>
<td>SRKXX-CPA(VDW)</td>
</tr>
<tr>
<td>111</td>
<td>Peng Robinson Ext. - CPA</td>
<td>PRX-CPA(VDW)</td>
</tr>
<tr>
<td>112</td>
<td>Peng Robinson Ext. - CPA + NRTL (Modified Huron Vidal)</td>
<td>PRXCPA-NRTL(P-HV)</td>
</tr>
<tr>
<td>113</td>
<td>Peng Robinson Ext. - CPA + NRTL (Modified LCVM)</td>
<td>PRXCPA-NRTL(P-LCVM)</td>
</tr>
<tr>
<td>114</td>
<td>Peng Robinson Ext. - CPA + NRTL (Modified MHV2)</td>
<td>PRXCPA-NRTL(P-MHV2)</td>
</tr>
<tr>
<td>115</td>
<td>Peng Robinson Ext. - CPA + NRTL (Modified Sandler)</td>
<td>PRXCPA-NRTL(P-WS)</td>
</tr>
<tr>
<td>120</td>
<td>Soave-Redlich_Kwong Ext. - CPA + NRTL (Modified Huron Vidal)</td>
<td>SRKXX-CPA-NRTL(P-HV)</td>
</tr>
<tr>
<td>130</td>
<td>UNIFAC</td>
<td>UNIFAC</td>
</tr>
<tr>
<td>150</td>
<td>Solid Pure (derived from)</td>
<td>SPRX-NRTL(P-HV)</td>
</tr>
<tr>
<td>151</td>
<td>Solid Pure (derived from)</td>
<td>SPRXCPA-NRTL(P-HV)</td>
</tr>
<tr>
<td>152</td>
<td>Solid Solution (derived from)</td>
<td>SSPRX-NRTL(P-HV)</td>
</tr>
<tr>
<td>170</td>
<td>Hydrate (derived from)</td>
<td>HP RXCPA-NRTL(P-HV)</td>
</tr>
<tr>
<td>171</td>
<td>Hydrate (derived from)</td>
<td>HP RX-NRTL(P-HV)</td>
</tr>
<tr>
<td>180</td>
<td>Wax</td>
<td>Wax</td>
</tr>
<tr>
<td>185</td>
<td>Asphaltene</td>
<td>Asphaltene</td>
</tr>
<tr>
<td>200</td>
<td>Pitzer (Electrolyte)</td>
<td>Pitzer</td>
</tr>
<tr>
<td>205</td>
<td>Peng Robinson Ext. - CPA-(MSA) + NRTL (Modified Huron Vidal)</td>
<td>PRXCPA-E-NRTL(P-HV)</td>
</tr>
<tr>
<td>210</td>
<td>P-SAFT-(MSA) Electrolyte</td>
<td>PSAFT-E</td>
</tr>
<tr>
<td>300</td>
<td>Steam tables based on IAPWS 1995 formulation</td>
<td>IAPWS 95</td>
</tr>
<tr>
<td>310</td>
<td>GERG 2008</td>
<td>GERG 2008</td>
</tr>
<tr>
<td>311</td>
<td>ISO 18453 (GERG)</td>
<td>ISO 18453</td>
</tr>
<tr>
<td>315</td>
<td>ISO 20765 (AGA 8)</td>
<td>ISO 20765</td>
</tr>
</tbody>
</table>
**Methods to define thermodynamic models**

To define or retrieve the thermodynamic models associated with a stream the library includes several methods. `setKM()` works with predefined packages while `setMP()`, `getMP()` allow to define specific models on each property (Fg, H, S, V...).

```plaintext
integer res = setKM (integer stream, integer Kcode)
given a stream and the code for the predefined package (contact Prode for the list of predefined packages available in different versions) sets the package.

integer res = setMP(integer stream, integer mp, integer state, integer model)
given a stream, property (Fg, H, S...) model and state (Vapor, Liquid, Solid, Hydrate) this method sets the specified model for that property and returns TRUE in case of success, otherwise returns FALSE.

integer m = getMP(integer stream, integer mp, integer state)
given a stream, related property (Fg, H, S...) and state (Vapor, Liquid, Solid, Hydrate) this method returns the specified model for that property and state.
```

**Table of codes to specify the different properties in setMP() and getMP()**

reference : methods setMP(), getMP() ...

<table>
<thead>
<tr>
<th>Code</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fugacity</td>
</tr>
<tr>
<td>2</td>
<td>Enthalpy</td>
</tr>
<tr>
<td>3</td>
<td>Entropy</td>
</tr>
<tr>
<td>4</td>
<td>Volume</td>
</tr>
<tr>
<td>5</td>
<td>Viscosity</td>
</tr>
<tr>
<td></td>
<td>(additional properties available in extended versions)</td>
</tr>
</tbody>
</table>

**Methods to define base values for Enthalpy and Entropy**

The library allows to define the base values (the temperature and initial value from which to start integration) for entropy and enthalpy from Properties Editor, in setting's page, these values are stored in archive and restored when program starts.

In addition it is possible to modify these value by code with the following methods,

```plaintext
integer res = setHB(integer mod, double t, double val)
given a code to identify the procedure (see the table with codes), the temperature and initial value sets base value for enthalpy.

integer res = setSB(integer mod, double t, double val)
given a code to identify the procedure (see the table with codes), the temperature and initial value sets base value for entropy.
```

**Table of codes to specify the different base values in setHB() and setSB()**

reference : methods setHB(), setSB() ...

<table>
<thead>
<tr>
<th>Code</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>initial values specified by user (values of t and val)</td>
</tr>
<tr>
<td>2</td>
<td>initial values are enthalpy of formation (or entropy of formation) and temperature 25 C</td>
</tr>
</tbody>
</table>
**Methods to set / access stream’s names**

In Prode Properties streams have several properties including a label (name) which could match (for example) the name of a line in your project, you can easily set / access these labels through a series of methods.

```
integer str = MStrN(integer stream, char *s, integer slm)
given a integer (that identifies a stream ) method fills string s with the name of stream (eventually truncated to slm maximum length), this is the Microsoft specific method
```

```
char *str = StrN(integer stream)
given a integer (that identifies a stream ) method returns as ANSI C type the string identifying that stream.
```

```
integer res = putN(integer stream, char *str)
given a integer (that identifies a stream ) and a ANSI C string identifying that stream this method sets the label.
```

**Methods to access Model’s data**

Prode Properties includes models for calculating properties as fugacities, enthalpies, entropies, volumes, viscosities etc. these methods allow to access the models available

```
integer nr = getMDnr()
returns the number of models available in this version
```

```
char *str = getMDN(int model)
given the model position (in the range 1-number of models available) method returns as ANSI C type the string identifying that model.
```

```
integer res = getMDP(int model, int prop, int state)
given the model position (in the range 1-number of models available) the required property and state returns TRUE if model can calculate the specified property, otherwise returns FALSE
```

```
integer code = getMDC(int model)
given the model position (in the range 1-number of models available) returns the code of the model
```
**Methods to control error's messages**

PROPERTIES includes a set of functions to control the error messages. By default PROPERTIES produces an error message via a Microsoft Windows Dialog Box every time an error is discovered. This approach can slow down the process when a long sequence of errors occurs in an extended calculation sequence, such as an iterative convergence calculation. A better solution in that situation is to provide a status flag that can be interrogated and used by the users at convenient points in the sequence.

`setErrFlag (integer state)`

given a Boolean (state) sets the error flag to TRUE or FALSE. The flag should be cleared (state = FALSE) before each sequence of calculations and tested (method `getErrFlag()` ) after the calcs. If this is done, then a flag state of TRUE indicates that an error has occurred somewhere in the calculation sequence).

`integer res = getErrFlag ()`

a value of TRUE means that an error has been found, please note that PROPERTIES doesn't clear the error flag state, You should clear the error flag (via `setErrFlag()` ) before each sequence of calc's.

`defErrMsg (integer state)`

a value TRUE for variable state sets on the Microsoft Windows Dialog Box and a message will appear every time an error is discovered. A value FALSE sets off the dialog box (no messages of error).

`integer str = MErrMsg(char *s, integer slm)`

fills string s with the last error message generated (eventually truncated to slm maximum length), this is the Microsoft specific method

`char *str = ErrMsg()`

Returns the last error message generated, this is the ANSI C compatible method

**Methods for accessing data-editing windows**

Prode Properties includes two predefined methods for activating Properties editor

`integer res = edS(integer stream)`

given a integer (that identifies a stream) method activates the Properties Editor on the specified stream

`integer res = edSS()`

this method activates the Properties Editor on first stream

**Methods to load / save archives**

Archives are files which contain a copy of the data used by Prode Properties to manage stream's and units of measurement, when you open an archive the stream's data and units are loaded, when you choose to save an archive these data are stored in a file. Archives are useful to create copies of your work otherways all data will be lost when leaving the application, Prode Properties includes methods for operations on archives.

`integer res = AOpen()`

open a file as archive (browse for file)

`integer res = AFOpen(char *path)`

open the file specified in *path as archive

`integer res = ASave()`

save a file as archive (browse for file)

`integer res = AFSave(char *path)`

save the file specified in *path as archive
Methods for accessing / defining the units of measurement

Prode Properties includes methods for accessing and defining the units of measurement, these methods utilize a numeric code for identifying the correspondent quantities. Refer to the paragraph "Access via software to the units of measurement" for a list of these codes.

```c
integer res = getUMC(integer UM)
given a integer (that identifies a quantity) method returns the selected UM for that quantity.

integer res = setUMC(integer UM, integer sel)
given two integers (the first identifies a quantity and the second the selection) method selects a UM for that quantity.

integer res = getUMN(integer UM)
given a integer (that identifies a quantity) method returns the number of different units of measurement available for that quantity.

integer str = MgetUMS(integer UM, integer sel, char *s, integer slm)
given two integers (the first identifies a quantity and the second the selection) fills string s with selected UM (eventually truncated to slm maximum length), this is the Microsoft specific method.

char *str = getUMS(integer UM, integer sel)
given two integers (the first identifies a quantity and the second the selection) method returns as ANSI C type the string identifying the selected UM.

integer str = MgetSUMS(integer UM, char *s, integer slm)
given a integer UM for quantity fills string s with selected UM (eventually truncated to slm maximum length), this is the Microsoft specific method.

char *str = getSUMS(integer UM)
given a integer UM for quantity this method returns as ANSI C type the string identifying the selected UM.

double res = UMCR(double value, integer UM, integer SEL)
given a value, the code for quantity and selection converts to reference and returns the result.

double res = UMCS(double value, integer UM, integer SEL)
given a value, the code for quantity and selection converts from reference and returns the result.

integer res = UMAU(double a, double b, char *name, integer UM)
given the code for a quantity, the parameters a, b required for conversion and the name adds a new (user defined, temporary) unit.

integer res = UMRAU(integer UM)
given the code for a quantity removes all additional (temporary) units.

Additional methods

double p = getPatm()
returns the internal reference (user defined) for atmospheric pressure quantity.
Application examples

We present here some notes about Prode Properties applications in form of FAQ that should assist users to easily extend features or add interfaces.

Tips on creation of Prode Properties applications

- Include a command (menu, button etc.) for accessing the Properties Editor (method edS(stream))
- Ensure that units of measurement are those defined in Prode Properties or include methods to set the units.
- Use isSDef() method to test a streams validity before accessing the stream. Accessing an undefined stream generates a large numbers of errors.
- Include functions for controlling error messages if you have extended calculation sequences. When managing error messages ensure that you test at the end of the calculation sequence to capture any problems that may have occurred.
- When debugging always attempt to limit the complexity of problems and expand progressively to the full application, retesting at intervals as you expand the scope of your problem.

User defined ID for accessing the components in chemical's file

In Prode Properties each component in chemical's file includes a ID which permits to access that component, this feature permits to maintain a unique identification number also when the chemical file changes. The ID must be a integer type, as default the CAS number has been adopted but the user may edit and change this value from the Properties Editor. The functions for accessing components in Prode Properties require the component code, this code may vary in different versions of chemical file, to convert the ID to the component code utilize the method CompCID() and CompID() to convert from code to ID.

```
' this example shows how to access data
' with ID code
Dim code As Long, ID As Long, Pc As Double
ID = 74840 ' CAS code for Ethane (but the user may define his own list of values)
code = CompCID(ID) ' get the code
Pc = CompPc(code) ' and the critical pressure
```
How to define directly a stream (without accessing the Properties Editor)

Prode Properties includes methods to access (read and write) each different value in a stream, making it possible for the user to create a procedure to define / edit / update directly each value without going through the Properties editor. Following list presents the methods for accessing all items:

- operating pressure, `getP()` `setOp()`
- operating temperature, `getT()` `setOp()`
- flow, `getW()` `setW()`
- vector [getMCNr() elements] with components codes, `getCC()` `putCC()`
- vector [getMCNr() elements] Z vector, mole basis, `getZ()` `putZ()`
- thermodynamic model (and related options), `getMP()` `setMP()`
- vector[getMBPNr() elements] of Ci, `getCi()` `putCi()`
- vector[getMBPNr() elements] of Cj, `getCj()` `putCj()`
- vector[getMBPNr() elements] of BIP matrix, `getBIP()` `putBIP()`

When defining a stream one must follow these steps:

- call `initS()` method to clear all stream's data
- define data
- call `setS()` method to validate the data

The following example shows how to define a 2 components stream:

```
Call initS(Stream)
Call setMP(Stream, Fg, SRK, 0)  ' see the paragraph "Codes used in Prode library"
Call setMP(Stream, Fg, SRK, 1)  ' define the models for all required properties and states
Call putZ(Stream, 1, z1)
Call putCC(Stream, 1, cc1)
Call putZ(Stream, 2, z2)
Call putCC(Stream, 2, cc2)
Call setS(Stream)  ' validates stream

Call setW(Stream, W)  ' flow

Call loadSB(Stream, 0)  ' load VLE BIPS from database
```
How to save and restore streams to / from a file

Archives are files which contain a copy of all compositions, operating conditions, units of measurement, settings etc., archives are useful for creating copies of your work otherwise all data will be lost when leaving the application.

The library includes methods to load and save archives, see the paragraph “Methods to load / save archives” for the list.

Call AOpen()  ' open a file as archive (browse for file)
Call AFOpen(“e:/def.ppp”)  ' open the specified file as archive
Call ASave()  ' save a file as archive (browse for file)
Call AFSave(“e:/def.ppp”)  ' save the specified file as archive
Error messages

PROPERTIES may generate the following error messages. For some of these, an action is suggested.

**Memory allocation error**
A limit in resources allocation (close applications, release memory and restart)

**Corrupted file, error reading data file**
PROPERTIES cannot access a file, this may depend from the file not being in the proper directory or being corrupted, it is suggested that You reinstall PROPERTIES.

**Internal error**
This error may depend from several different conditions, the most common is a wrong parameter in a function (i.e. an attempt to pass a value out of permitted range). Check Your code.

**too many local variables**
**too many variables**
a limit in resources allocation (see above)

**calc. on undefined stream data**
an undefined stream found while executing calc's (edit and define the stream)

**undefined stream's operating conditions**
pressure, temperature or flow are undefined (edit and define the stream)

**error calling thermo calc. procedure**
wrong input value (calcs cannot converge) or calcs outside temperature range (check chemical's file for limits in temperature correlation's).

**cannot converge calc's loop**
A wrong convergence condition has been specified (i.e in an adiabatic flash calcs a thermal condition that cannot be reached by varying temperature, pressure or liquid fraction ; a parameter is outside range limits etc.)

**T, P values outside H, S range calcs**
A wrong condition has been specified and a parameter in enthalpy /entropy calcs is outside range limits

**too many comp's in a stream**
when two or more streams are mixed the total nr. of components may exceed the maximum some inconsistencies in stream's data

**error accessing component's data archive**
unavailable data (a unspecified component) or calc's outside temperature range.

**Stack error (no memory), reload procedure**
a limit in resources allocation (see above)

**Method not available in this version**
Attempt to define a method not available in that version, edit the stream and define a new method

**A stream with Steam Tables model must have only 1 component**
You should specify a stream with one component only in order to apply ASME Steam Tables model
Calculation basis

The user can specify which method to use selecting the models. Please refer to the paragraph “reference literature” and “Models” for additional information about the methods.

Fugacity calculated according selected model
Enthalpy calculated according selected model
Entropy calculated according selected model
Volume calculated according selected model

Viscosity
- gas: low pressure mixing rule according Wilke (1950), operating conditions correction according Stiel and Thodos (1964).
- liquid: logarithmic average mixing rule, pressure correction according Lucas (1981)

Thermal conductivity
- gas: low pressure mixing rule according Mason and Saxena (1958), operating conditions correction according Stiel and Thodos (1964)
- liquid: mixing rule according Li (1976)

Surface tension
mixing rule according MacLeod-Sugden

Heat of combustion
weight average mixing rule according ISO std. (database contains values in Kj/Kg)

Flammability limits
mixing rule according Le Chatelier as discussed by Coward & Jones (1952)

Limits in thermodynamic calc’s

Enthalpy, Entropy calc’s
In Prode Properties the user can specify different initial conditions for enthalpy and entropy, see the paragraph “Config settings” for additional details.

Temperature, pressure ranges

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature range</td>
<td>1 K - 5000 K</td>
</tr>
<tr>
<td>Pressure range</td>
<td>1 Pa – 1000 Bar</td>
</tr>
</tbody>
</table>
### Chemical’s File format

Note: all data dependent correlations in chemicals file have a range of temperature for application, outside this range they may provide inconsistent results. Prode Properties checks for this range (as defined by high and low limits in chemicals file) and attempts to extend data when required (when operating conditions are outside the range of application of correlations), this may produce in some cases inconsistent results with simple models and properties which require differentiation, for example specific heat capacity.

### Flexible data format

Prode Properties utilizes proprietary code which allows up to 30 correlations and custom units to define each temperature dependent property, all major standards including DIPPR and others are supported.

### Chemical's data file

Prode Properties base version adopts the following format

<table>
<thead>
<tr>
<th>Property</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula string</td>
<td>12 chars max</td>
</tr>
<tr>
<td>Name (1) (main list)</td>
<td>string 40 chars max</td>
</tr>
<tr>
<td>Name (2) (user defined list)</td>
<td>string 40 chars max</td>
</tr>
<tr>
<td>Name (3) (user defined list)</td>
<td>string 40 chars max</td>
</tr>
<tr>
<td>Identification number (CAS as default)</td>
<td></td>
</tr>
<tr>
<td>Molecular weight</td>
<td></td>
</tr>
<tr>
<td>Critical temperature</td>
<td></td>
</tr>
<tr>
<td>Critical pressure</td>
<td></td>
</tr>
<tr>
<td>Critical volume</td>
<td></td>
</tr>
<tr>
<td>Acentric factor</td>
<td></td>
</tr>
<tr>
<td>Dipole Moment</td>
<td></td>
</tr>
<tr>
<td>Radius of Gyration</td>
<td></td>
</tr>
<tr>
<td>Solubility parameter</td>
<td></td>
</tr>
<tr>
<td>Standard enthalpy of formation (298 K)</td>
<td></td>
</tr>
<tr>
<td>Gibbs free energy of formation (298 K, 1 atm)</td>
<td></td>
</tr>
<tr>
<td>Enthalpy of fusion</td>
<td></td>
</tr>
<tr>
<td>Normal boiling point</td>
<td></td>
</tr>
<tr>
<td>Melting point</td>
<td></td>
</tr>
<tr>
<td>Flammability lean limit % (range 0-100)</td>
<td></td>
</tr>
<tr>
<td>Flammability rich limit % (range 0-100)</td>
<td></td>
</tr>
<tr>
<td>Autoignition temperature</td>
<td></td>
</tr>
<tr>
<td>Net heat of combustion</td>
<td></td>
</tr>
<tr>
<td>Flash Point</td>
<td></td>
</tr>
<tr>
<td>Gas heat capacity correlation</td>
<td></td>
</tr>
<tr>
<td>type of equation</td>
<td></td>
</tr>
<tr>
<td>unit for property</td>
<td></td>
</tr>
<tr>
<td>unit for temperature</td>
<td></td>
</tr>
<tr>
<td>low temperature limit</td>
<td></td>
</tr>
<tr>
<td>high temperature limit</td>
<td></td>
</tr>
<tr>
<td>A-E (5 parameters)</td>
<td></td>
</tr>
<tr>
<td>Vapor viscosity correlation</td>
<td></td>
</tr>
<tr>
<td>type of equation</td>
<td></td>
</tr>
<tr>
<td>unit for property</td>
<td></td>
</tr>
<tr>
<td>unit for temperature</td>
<td></td>
</tr>
<tr>
<td>low temperature limit</td>
<td></td>
</tr>
<tr>
<td>high temperature limit</td>
<td></td>
</tr>
<tr>
<td>A-E (5 parameters)</td>
<td></td>
</tr>
<tr>
<td>Vapor thermal conductivity correlation</td>
<td></td>
</tr>
<tr>
<td>type of equation</td>
<td></td>
</tr>
<tr>
<td>unit for property</td>
<td></td>
</tr>
<tr>
<td>unit for temperature</td>
<td></td>
</tr>
<tr>
<td>low temperature limit</td>
<td></td>
</tr>
<tr>
<td>high temperature limit</td>
<td></td>
</tr>
<tr>
<td>A-E (5 parameters)</td>
<td></td>
</tr>
</tbody>
</table>
Heat of vaporization correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Liquid vapor pressure correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Surface tension
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Liquid density correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Liquid viscosity correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Liquid thermal conductivity correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Liquid heat capacity correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Solid vapor pressure correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)
Solid density correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Solid thermal conductivity correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)

Solid heat capacity correlation
  type of equation
  unit for property
  unit for temperature
  low temperature limit
  high temperature limit
  A-E (5 parameters)
Sources of data

Data in chemical data file come from several sources including:
- “Dechema Chemistry Data ser.” text books
- “DIPPR data collection” text books
- “Technical Data Book, Petroleum Refining”

Due to the large differences in critical and transport properties found in different sources, DIPPR (AICHE Design Institute for Physical Property Data) reference has been selected as a default.

Component's identification

Components are identified by name (from DIPPR list), chemical formula and Identification number.

Regression procedures and results

Coefficients in correlations have been calculated with a custom program that uses a modified version of Levenberg-Marquardt algorithm, reported errors (at each fitting point) are usually lower than 1% of input values for the most complex correlations (i.e. vapor pressure), however in some cases they may be higher.

Consistency tests

When relations exist between thermodynamic properties (i.e. acentric factor and critical pressure and temperature, vapor pressure and heat of vaporization etc.) a consistency test has been performed.

Comparing Prode Properties results against those of different process simulators

When comparing data from different tools one must verify that
- the different tools do use the same thermodynamic models
- properties in databanks have similar values
- lists and values of BIPs and other parameters which can influence results have similar values
Models

Prode Properties includes a complete set of thermodynamic models, see also the table “Features available vs. Versions”

Liquid activities
Wilson
NTRL
UNIQUAC

Predictive
UNIFAC

Electrolytes
Pitzer
CPA-electrolyte
SAFT-electrolyte

Cubic EOS
Soave-Redlich-Kwong, Peng-Robinson with std. alpha function and VdW mixing rules,
Extended versions of SRK and PR including parameters calculated to fit experimental data (saturation pressures, densities, heat capacities etc.) and different mixing rules to combine equations of state with activity models
Std. and Modified versions of Huron Vidal (HV) rule
Std. and Modified versions of Linear Combination of Vidal and Michelsen (LCVM) rule
Std. and Modified versions of Michelsen-Huron-Vidal (MHV2) rule
Std. and Modified versions of Wong Sandler (WS) rule
etc...

Other models
Modified Benedict-Webb-Rubin
Benedict-Webb-Rubin-Starling
Lee-Kesler
Lee-Kesler-Plocker

Models based on associating fluid theory
Different versions of CPA Cubic Plus Association based on Soave Redlich Kwong and Peng Robinson models with VdW mixing rules and several others to combine equations of state with activity models
Std. and Modified versions of Huron Vidal (HV) rule
Std. and Modified versions of Linear Combination of Vidal and Michelsen (LCVM) rule
etc...

Different versions of SAFT (Perrturbed Chain Statistical Associating Fluid Theory)

SOLIDS
SPM (Solid Pure Model) solid phase treated as single component
SSM (Solid Solution Model) solid phase treated as homogeneous solution
WAX solid phase treated as homogeneous solution (with specific parameters)
Asphaltene
Hydrates (based on Van der Waals and Plateeuw theory with a std. model and a complex model)

STANDARDS
GERG 2008 (ISO 20765)
AGA 2017 (2017 version with GERG 2008 formulations)
Steam tables (IAPWS 1995) Water / steam properties calculated according IAPWS 1995 formulation

The models export derivatives of Fg, H, S, V vs. W, P, T
**UNIFAC functional groups**

The underlying idea in UNIFAC method is that a molecule can be considered as a collection of functional groups. The main advantage of this approach is that from a relatively small number of functional groups the properties of many different molecules can be predicted. The UNIFAC model is useful for estimating solution behaviour in the absence of experimental data. Prode Properties incorporates the UNIFAC Group Contribution revision 5 (January 1992, J.P.Baker).

Following the main groups and subgroups table:

<table>
<thead>
<tr>
<th>Code</th>
<th>Main</th>
<th>Subgroup</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CH2</td>
<td>CH3</td>
<td>Hexane</td>
</tr>
<tr>
<td>2</td>
<td>CH2</td>
<td>n-Hexane</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CH</td>
<td>2-Methylpropane</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>Neopentane</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>C=C</td>
<td>CH2=CH</td>
<td>1-Hexene</td>
</tr>
<tr>
<td>6</td>
<td>CH=CH</td>
<td>2-Hexene</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CH2=C</td>
<td>2-Methyl-1-butene</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CH=C</td>
<td>2-Methyl-2-butene</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>C=C</td>
<td>2,3-Dimethylbutene</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ACH</td>
<td>ACH</td>
<td>Naphthaline</td>
</tr>
<tr>
<td>10</td>
<td>AC</td>
<td>Styrene</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ACCH2</td>
<td>ACCH3</td>
<td>Toluene</td>
</tr>
<tr>
<td>12</td>
<td>ACCH2</td>
<td>EthylBenzene</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>ACCH</td>
<td>Cumene</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>OH</td>
<td>OH</td>
<td>n-Propanol</td>
</tr>
<tr>
<td>15</td>
<td>CH3OH</td>
<td>CH3OH</td>
<td>Methanol</td>
</tr>
<tr>
<td>16</td>
<td>H2O</td>
<td>H2O</td>
<td>Water</td>
</tr>
<tr>
<td>17</td>
<td>ACOH</td>
<td>ACOH</td>
<td>Phenol</td>
</tr>
<tr>
<td>18</td>
<td>CH2CO</td>
<td>CH3CO</td>
<td>Butanone</td>
</tr>
<tr>
<td>19</td>
<td>CH2CO</td>
<td>Pentanone-3</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>CHO</td>
<td>CHO</td>
<td>Propionic aldehyde</td>
</tr>
<tr>
<td>21</td>
<td>CCOO</td>
<td>CH3COO</td>
<td>Butyl acetate</td>
</tr>
<tr>
<td>22</td>
<td>CH2COO</td>
<td>Methyl propionate</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>HCOO</td>
<td>HCOO</td>
<td>Ethyl formate</td>
</tr>
<tr>
<td>24</td>
<td>CH2O</td>
<td>CH3O</td>
<td>Dimethyl ether</td>
</tr>
<tr>
<td>25</td>
<td>CH2O</td>
<td>Diethyl ether</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>CHO</td>
<td>Diisopropyl ether</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>THF</td>
<td>Tetrahydrofuran</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>CNH2</td>
<td>CH3NH2</td>
<td>Methylamine</td>
</tr>
<tr>
<td>29</td>
<td>CH2NH2</td>
<td>Ethyl amine</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>CH3NH2</td>
<td>Isopropylamine</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>CNH</td>
<td>CH3NH</td>
<td>Dimethylamine</td>
</tr>
<tr>
<td>32</td>
<td>CH2NH</td>
<td>Diethyl amine</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>CHNH</td>
<td>Diisopropylamine</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>(C)3N</td>
<td>CH3N</td>
<td>Trimethylamine</td>
</tr>
<tr>
<td>35</td>
<td>CH2N</td>
<td>CH2N</td>
<td>Triethylamine</td>
</tr>
<tr>
<td>36</td>
<td>ACNH2</td>
<td>ACNH2</td>
<td>Aniline</td>
</tr>
<tr>
<td>37</td>
<td>Pyridine</td>
<td>CSH5N</td>
<td>Pyridine</td>
</tr>
<tr>
<td>38</td>
<td>CSH4N</td>
<td>2-Methyl pyridine</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>CSH3N</td>
<td>2,3-Dimethylpyridine</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>CCN</td>
<td>CH3CN</td>
<td>Acetonitrile</td>
</tr>
<tr>
<td>41</td>
<td>CH2CN</td>
<td>Propionitrile</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>COOH</td>
<td>COOH</td>
<td>Acetic acid</td>
</tr>
<tr>
<td>43</td>
<td>HCOOH</td>
<td>Formic acid</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>CCl</td>
<td>CH2Cl</td>
<td>Butane-1-chloro</td>
</tr>
<tr>
<td>45</td>
<td>CHCl</td>
<td>Propane-2-chloro</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>CCl</td>
<td>CCl</td>
<td>2-Methylpropane-2-chloro</td>
</tr>
<tr>
<td>47</td>
<td>CCl2</td>
<td>CH2Cl2</td>
<td>Methane-dichloro</td>
</tr>
<tr>
<td>48</td>
<td>CHCl2</td>
<td>Ethane-1,1-dichloro</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>CCl2</td>
<td>Propane-2,2-dichloro</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>CCl3</td>
<td>CHCl3</td>
<td>Chloroform</td>
</tr>
<tr>
<td>51</td>
<td>CCl3</td>
<td>CCl3</td>
<td>Ethane-1,1,1-trichloro</td>
</tr>
<tr>
<td>52</td>
<td>CCl4</td>
<td>CH4</td>
<td>Methane-tetrachloro</td>
</tr>
<tr>
<td>53</td>
<td>ACCl</td>
<td>ACCl</td>
<td>Benzene-chloro</td>
</tr>
<tr>
<td>54</td>
<td>CNO2</td>
<td>CH3NO2</td>
<td>NitroMethane</td>
</tr>
<tr>
<td>55</td>
<td>CH2NO2</td>
<td>Propane-1-nitro</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>CHNO2</td>
<td>Propane-2-nitro</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Main</td>
<td>Subgroup</td>
<td>Example</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>57</td>
<td>ACNO2</td>
<td>ACNO2</td>
<td>Benzene-nitro</td>
</tr>
<tr>
<td>58</td>
<td>CS2</td>
<td>CS2</td>
<td>Carbon Disulfide</td>
</tr>
<tr>
<td>59</td>
<td>CH3SH</td>
<td>CH3SH</td>
<td>Methanethiol</td>
</tr>
<tr>
<td>60</td>
<td>CH2SH</td>
<td>CH2SH</td>
<td>Ethanethiol</td>
</tr>
<tr>
<td>61</td>
<td>Furfural</td>
<td>Furfural</td>
<td>Furfural</td>
</tr>
<tr>
<td>62</td>
<td>DOH</td>
<td>DOH</td>
<td>1,2-Ethanediol</td>
</tr>
<tr>
<td>63</td>
<td>I</td>
<td>I</td>
<td>Iodoethane</td>
</tr>
<tr>
<td>64</td>
<td>Br</td>
<td>Br</td>
<td>Bromoethane</td>
</tr>
<tr>
<td>65</td>
<td>C-C</td>
<td>CH-C</td>
<td>Hexyne-1</td>
</tr>
<tr>
<td>66</td>
<td>C-C</td>
<td>CH-C</td>
<td>Hexyne-2</td>
</tr>
<tr>
<td>67</td>
<td>DMSO</td>
<td>DMSO</td>
<td>Dimethylsulfoxide</td>
</tr>
<tr>
<td>68</td>
<td>ACRY</td>
<td>ACRY</td>
<td>Acrylnitrile</td>
</tr>
<tr>
<td>69</td>
<td>CICC</td>
<td>Cl-(C=C)</td>
<td>Ethene-trichloro</td>
</tr>
<tr>
<td>71</td>
<td>ACF</td>
<td>ACF</td>
<td>Hexafluorobenzene</td>
</tr>
<tr>
<td>72</td>
<td>DMF</td>
<td>DMF-1</td>
<td>N,N-Dimethylformamide</td>
</tr>
<tr>
<td>73</td>
<td>DMF</td>
<td>DMF-2</td>
<td>N,N-Diethylformamide</td>
</tr>
<tr>
<td>74</td>
<td>CF2</td>
<td>CF3</td>
<td>Perfluorohexane</td>
</tr>
<tr>
<td>75</td>
<td>CF2</td>
<td>CF3</td>
<td>Perfluorohexane</td>
</tr>
<tr>
<td>76</td>
<td>CF</td>
<td>CF</td>
<td>Perfluoromethylcyclohexane</td>
</tr>
<tr>
<td>77</td>
<td>COO</td>
<td>COO</td>
<td>Methyl acrylate</td>
</tr>
<tr>
<td>78</td>
<td>SiH2</td>
<td>SiH3</td>
<td>Methylsilane</td>
</tr>
<tr>
<td>79</td>
<td>SiH2</td>
<td>SiH3</td>
<td>Diethylsilane</td>
</tr>
<tr>
<td>80</td>
<td>SiH</td>
<td>SiH</td>
<td>Heptamethyltrisiloxane</td>
</tr>
<tr>
<td>81</td>
<td>Si</td>
<td>Si</td>
<td>Heptamethyltrisiloxane</td>
</tr>
<tr>
<td>82</td>
<td>SiO</td>
<td>SiH2O</td>
<td>1,3-Dimethyltrisiloxane</td>
</tr>
<tr>
<td>83</td>
<td>SiO</td>
<td>SiH2O</td>
<td>1,1,3,3-Tetramethyltrisiloxane</td>
</tr>
<tr>
<td>84</td>
<td>SiO</td>
<td>SiO</td>
<td>Octamethylcyclotetrasiloxane</td>
</tr>
<tr>
<td>85</td>
<td>NMP</td>
<td>NMP</td>
<td>N-methylpyrrolidone</td>
</tr>
<tr>
<td>86</td>
<td>CClF</td>
<td>CCl3F</td>
<td>Trichlorofluoromethane</td>
</tr>
<tr>
<td>87</td>
<td>CCl2F</td>
<td>CCl2F</td>
<td>Tetrachloro-1,2-difluoroethane</td>
</tr>
<tr>
<td>88</td>
<td>HCl2F</td>
<td>HCl2F</td>
<td>Dichlorofluoromethane</td>
</tr>
<tr>
<td>89</td>
<td>HCClF</td>
<td>HCClF</td>
<td>1-Chloro-1,2,2,2-tetrafluoroethane</td>
</tr>
<tr>
<td>90</td>
<td>CClF2</td>
<td>CClF2</td>
<td>1,2-Dichlorotetrafluoroethane</td>
</tr>
<tr>
<td>91</td>
<td>HCClF</td>
<td>HCClF</td>
<td>Chlorodifluoromethane</td>
</tr>
<tr>
<td>92</td>
<td>CClF3</td>
<td>CClF3</td>
<td>Chlorotrifluoromethane</td>
</tr>
<tr>
<td>93</td>
<td>CClF2</td>
<td>CClF2</td>
<td>Dichlorodifluoromethane</td>
</tr>
<tr>
<td>94</td>
<td>CON</td>
<td>CONH2</td>
<td>Acetamid</td>
</tr>
<tr>
<td>95</td>
<td>CON</td>
<td>CONH3</td>
<td>N-Methylacetamid</td>
</tr>
<tr>
<td>96</td>
<td>CON</td>
<td>CONH2</td>
<td>N-Ethylacetamid</td>
</tr>
<tr>
<td>97</td>
<td>CON</td>
<td>CONH2</td>
<td>N,N-Dimethylacetamid</td>
</tr>
<tr>
<td>98</td>
<td>CON</td>
<td>CONH3</td>
<td>N,N-Dimethylacetamid</td>
</tr>
<tr>
<td>99</td>
<td>CON</td>
<td>CONH2</td>
<td>N,N-Diethylacetamid</td>
</tr>
<tr>
<td>100</td>
<td>OCCOH</td>
<td>C2H5O2</td>
<td>2-Ethoxyethanol</td>
</tr>
<tr>
<td>101</td>
<td>CH2S</td>
<td>CH2S</td>
<td>Dimethylsulfide</td>
</tr>
<tr>
<td>102</td>
<td>C4H4S</td>
<td>C4H4S</td>
<td>Thiophene</td>
</tr>
<tr>
<td>103</td>
<td>CH2S</td>
<td>CH2S</td>
<td>Diethylsulfide</td>
</tr>
<tr>
<td>104</td>
<td>CHS</td>
<td>CHS</td>
<td>Diisopropylsulfide</td>
</tr>
<tr>
<td>105</td>
<td>Morpholine</td>
<td>MORPH</td>
<td>Morpholine</td>
</tr>
<tr>
<td>106</td>
<td>Thiophene</td>
<td>C4H4S</td>
<td>Thiophene</td>
</tr>
<tr>
<td>107</td>
<td>C4H3S</td>
<td>C4H3S</td>
<td>2-Methylthiophene</td>
</tr>
<tr>
<td>108</td>
<td>C4H2S</td>
<td>C4H2S</td>
<td>2,3-Dimethylthiophene</td>
</tr>
</tbody>
</table>